Adaptation des services sensibles au contexte selon une approche intentionnelle

Najar Salma

THESE DE DOCTORAT

DE L'UNIVERSITE DE PARIS I – PANTHEON-SORBONNE

Spécialité : Informatique

Salma Najar

Pour l'obtention du titre de :

DOCTEUR DE L'UNIVERSITE PARIS I – PANTHEON-SORBONNE

Adaptation des services sensibles au contexte selon une approche intentionnelle

Soutenu le 11 Avril 2014 devant le jury composé de :

Mme Carine SOUVEYET	Directeur de Thèse
Mme Manuele KIRSCH-PINHEIRO	Co-Directeur de Thèse
Mme Isabelle MIRBEL	Rapporteur
Mr Jérôme GENSEL	Rapporteur
Mme Corine CAUVET	Membre de Jury
Mme Bénédicte LE-GRAND	Membre de Jury

« Chercher n'est pas une chose et trouver une autre,
mais le gain de la recherche, c'est la recherche même. »
de Saint Grégoire de Nysse

A mes parents, mon amour sans faille,
A Sana et Omar, ma sœur et mon frère adorés,
A Aymen, mon amour et ma source d'inspiration

Remerciements

La thèse est une aventure extraordinaire qui nous pousse au bout de nous-mêmes et qui nous construit... C'est une belle expérience, difficile par moment et agréable par d'autres… Je me suis sentie complètement transformée et grandie par cette expérience… J'ai vécu mon doctorat comme une véritable aventure, un enrichissement personnel et professionnel considérable…C'est l'aboutissement d'une étape importante dans ma vie et le commencement d'une autre que j'espère aussi attractive… Un énorme Merci à toutes les personnes qui ont contribué pour que cette expérience soit si exceptionnelle.

Je tiens à remercier en premier lieu, Carine Souveyet, Professeur à l'Université Paris 1 Panthéon - Sorbonne pour m'avoir accordée sa confiance en acceptant de diriger mes recherches. Je la remercie infiniment de m'avoir fait bénéficier tout au long de ce travail de sa grande compétence, de sa rigueur intellectuelle et de ses précieux conseils. Sa revendication constante du travail sérieux m'a aidée à progresser. Que le fruit de ces longues années de travail soit à la hauteur de ce qu'elle a semé en moi.

J'exprime également toute ma gratitude à Manuele Kirsch-Pinheiro, Maître de Conférence à l'Université de Paris 1 Panthéon - Sorbonne qui m'a épaulée tout au long de la réalisation de ce travail de recherche et prodiguée de précieux conseils qui m'ont permis d'aller constamment de l'avant. Elle m'a encouragée par ses orientations sans cesser d'être une grande source de motivation et de persévérance. Son exigence m'a permise de progresser dans l'élaboration de ma recherche. Qu'elle trouve dans ce travail l'expression de ma profonde et sincère reconnaissance.

Je souhaiterais adresser mes remerciements les plus sincères à Isabelle Mirbel, Maître de Conférence Habilitée à Diriger des Recherches à l'Université de Nice Sophia Antipolis, et Jérôme Gensel, Professeur à l'Université de Pierre Mendès France - Grenoble 2, pour m'avoir fait l'honneur d'être les rapporteurs de ce travail. Je les remercie pour leur application et leurs précieux commentaires et recommandations.

Je voulais également remercier les examinateurs de ce travail, Madame Corine Cauvet, Professeur à l'Université d'Aix-Marseille et Madame Bénédicte Le Grand, Professeur à l'Université de Paris 1 Panthéon – Sorbonne pour avoir accepté de faire partie de mon jury de thèse et d'évaluer mon travail. Je remercie profondément Bénédicte qui a eu la gentillesse de me consacrer du temps pour échanger avec moi autour de mon sujet de thèse. Ses remarques et ses conseils m'ont été d'une aide précieuse.

J'ai pu travailler dans un cadre particulièrement agréable, grâce à l'ensemble des membres de l'équipe du Centre de Recherche en informatique (CRI). J'exprime toute ma gratitude envers tous ceux qui m'ont aidée et encouragée tout au long de mon séjour au sein de l'équipe et surtout l'ensemble des thésards pour leur enthousiasme et leur soutien

quotidien. Merci à tous pour votre bonne humeur, pour toutes ces séances de rires et de sourires, et pour toutes ces discussions autour d'un café.

Cet aboutissement n'aurait jamais pu se faire sans l'encouragement, le soutien et l'amour de tous les membres de ma famille et de mes amis. A mes chers parents, à ma sœur Sana et à mon frère Omar : Merci d'avoir cru en moi… Merci pour votre patience et pour l'affection que vous m'avez manifestée durant ces années… A ma belle famille : Merci du fond du cœur pour votre soutien et pour vos précieux conseils… A mes amis : Merci d'être là et de m'avoir toujours soutenue et m'avoir changée les idées quand j'en avais besoin … Une dédicace toute particulière à Sonda et Fatma, deux amies très spéciales qui m'ont soutenue dès le début et m'ont aidée dans les périodes de doute.

Je garde pour la fin un remerciement particulier pour mon mari, Aymen. Je te remercie infiniment de m'avoir aidée, soutenue, encouragée et surtout aimée... Ces dernières années n'ont pas été si simples, tu m'as supportée dans les périodes les plus difficiles… Tu m'as aidée à garder le moral haut et surtout à surmonter les difficultés. Merci d'avoir été toujours là…

Pour finir, je souhaiterais dédier ce travail à tous ceux que j'aime. Qu'ils y trouvent ici l'expression de ma profonde affection et de mes plus sincères remerciements.

Résumé

L'émergence des nouvelles technologies a fait évoluer l'usage de ces technologies dans la perspective d'accéder aux différents systèmes qui prennent place dans notre vie quotidienne à n'importe quel endroit et à tout moment. En effet, la démocratisation des dispositifs et l'évolution des technologies mobiles ont bouleversé la manière dont on utilise ces systèmes dans un environnement pervasif. Dans le cadre de l'entreprise, ces nouvelles technologies ont élargi les frontières des Systèmes d'Information (SI) bien au-delà des frontières physiques de l'organisation. Les directeurs des Systèmes d'Information (DSI) sont ainsi confrontés à de nouveaux modes d'interaction entre le SI et son environnement. Les Systèmes d'Information sont ainsi confrontés à un environnement pour lequel ils n'ont pas été particulièrement conçus. Ces systèmes doivent maintenant faire face à un environnement pervasif, et à l'avenir, intégrer des éléments physiques ainsi que logiques et organisationnels. Nous assistons donc aujourd'hui à l'émergence d'une nouvelle génération de Systèmes d'Information : les « Systèmes d'Information Pervasifs » (SIP).

Les Systèmes d'Information Pervasifs se veulent alors une réponse à cette importante évolution des SI. Par contre, ils se doivent de gérer l'hétérogénéité et le dynamisme de l'environnement de manière transparente afin de satisfaire au mieux les besoins des utilisateurs. Nous ainsi sommes face à un problème de conception et de réalisation d'un SIP répondant à tous les besoins de transparence, d'adaptation à l'environnement et d'adaptation à l'utilisateur d'un SIP. Or les SIP constituent aujourd'hui une nouvelle génération des SI qui est difficile à conceptualiser, avec peu de méthodes et de modèles disponibles. Ainsi, il devient essentiel de mettre en place un cadre plus formel permettant d'aider les concepteurs à mieux comprendre les SIP et surtout à mieux les maîtriser, tout en assurant la transparence nécessaire à ces systèmes.

Dans ce travail de thèse, nous proposons une nouvelle vision intentionnelle et contextuelle des SIP. Cette nouvelle vision représente une vision centrée utilisateur d'un SIP transparent, non intrusif et compréhensible à l'utilisateur. Elle se base sur l'orientation service, la sensibilité au contexte et sur une approche intentionnelle afin de résoudre les problèmes de transparence, d'adaptation à l'environnement et d'adaptation aux utilisateurs. Par la suite, nous proposons une solution plus globale pour concrétiser cette vision intentionnelle et contextuelle des SIP. Nous proposons un cadre conceptuel des SIP décrivant et formalisant l'ensemble de ses éléments afin d'aider la DSI dans sa conception du système. Nous proposons ensuite des mécanismes de découverte et de prédiction de services qui sont intégrés dans une architecture de gestionnaire de SIP qui est conforme à ce cadre conceptuel. Finalement, nous proposons une démarche méthodologique de conception et de réalisation d'un SIP qui supporte le passage entre le cadre conceptuel et l'implémentation de l'architecture proposée.

Abstract

The emergence of new technologies has changed the use of these technologies in order to access to the various systems that take place in our daily life anywhere and anytime. Indeed, the democratization of devices and the evolution of mobile technologies have changed the way these systems are used in a pervasive environment. As part of the company, these new technologies have expanded the boundaries of Information Systems (IS) beyond the physical boundaries of the organization. Chiefs Information Officers (CIOs) are confronted with new modes of interaction with their SI. Information Systems are thus faced with an environment for which they were not specifically designed. These systems must now face a pervasive environment, and in the future, integrate physical as well as logical and organizational elements. We are now witnessing the emergence of a new generation of Information Systems: « Pervasive Information Systems« (PIS).

The Pervasive Information Systems want an answer to this important evolution of SI. Against, they must manage the heterogeneity and dynamism of the environment in a transparent manner to best meet the users needs. Thus, currently we are facing a problem of design and implementation of PIS that meets all the needs of transparency, adaptation to the environment and adaptation to the user. Although, PIS represent today a new generation of SI that is difficult to conceptualize, with few methods and models available. Thus, it becomes essential to establish a more formal framework to help designers better understand PIS and especially to the better control of it, while ensuring the necessary transparency to these systems.

In this thesis, we propose a new intentional and contextual vision of PIS. This new vision represents a user centric vision of PIS, which is transparent, non-intrusive and understandable to the user. It is based on service orientation, context-awareness and intentional approach to solve the problems of transparency, adaptation to the environment and adaptation to users. Subsequently, we propose a more comprehensive solution to achieve this intentional and contextual vision of PIS. We propose a conceptual framework for describing and formalizing PIS and all its elements to help CIOs in their system design. We then propose mechanisms for services discovery and prediction that are integrated in the architecture of PIS manager. Finally, we propose a methodological approach for the design and the implementation of PIS that supports the transition between the conceptual framework to its implementation with the proposed architecture.

Table des matières

Chapitre 1. INTRODUCTION GENERALE

Le travail réalisé dans le cadre de cette thèse se situe dans le domaine de l'*Ingénierie des Services*, de l'*Informatique Ubiquitaire (ou Pervasive)* et de l'*ingénierie des Systèmes d'Information*. Il présente une approche centrée utilisateur permettant de concevoir et de construire la nouvelle génération des Systèmes d'Information, qu'on appelle *Systèmes d'Information Pervasifs*.

1.1. CONTEXTE DE RECHERCHE : LA VISION D'UN SYSTEME D'INFORMATION PERVASIF

1.1.1. Informatique Pervasive

Depuis plusieurs années, nous assistons à l'émergence des nouvelles technologies. Nous sommes témoins de l'évolution de l'usage de ces technologies afin d'accéder aux différents systèmes qui prennent place dans notre quotidien. En effet, la démocratisation des dispositifs utilisés dans notre vie quotidienne (*Smartphones*, tablettes, etc.), ainsi que l'évolution des technologies mobiles (3G, géolocalisation, *Bluetooth*, etc.) et des autres technologies ont bouleversé la manière dont on utilise ces systèmes. Selon Musolesi (Musolesi, 2011), les *Smartphones*, par exemple, sont considérés comme la plateforme informatique de l'avenir, où l'informatique peut être fortement décentralisée et répartie sur les différents terminaux utilisés par les utilisateurs pour fournir des solutions hautement évolutives.

Ces avancées technologiques, qui nous donnent le pouvoir d'interagir avec le monde d'une manière naturelle, ont contribué à l'amélioration de nos capacités quotidiennes, fournissant, ainsi, des avantages à long-terme pour la société dans son ensemble (Cheng, 2010). Grâce à ces avancées, l'informatique s'est intégrée à l'environnement d'une façon invisible. Schmidt (Schmidt, 2010) illustre ceci par l'usage du GPS dans différentes situations et d'une manière presque inconsciente. Plus précisément, il utilise trois récepteurs GPS différents : le premier dans son téléphone, fournissant aux réseaux sociaux sa localisation, le second intégré dans le système de navigation de la voiture et le dernier dans l'appareil photo, afin de géo-localiser les photos prises. Cet auteur démontre à travers son exemple, que cette technologie est devenue invisible, comme d'autres technologies intégrées à notre environnement.

Cette réalité a été soutenue depuis plusieurs années par l'Informatique Ubiquitaire (ou Pervasive) (Weiser, 1991). Weiser (Weiser, 1991) soutient que « *les technologies les plus profondément enracinées sont les technologies invisibles. Elles s'intègrent dans la trame de la vie quotidienne jusqu'à ne plus pouvoir en être distinguées* ».

A l'instar de Bell et Dourish (Bell et Dourish, 2007), nous pensons que l'Informatique Ubiquitaire est déjà une réalité et non pas un avenir proche. Elle fait partie intégrante de notre quotidien, notamment à travers les nouvelles technologies. Celles-ci sont devenues presque invisibles à nos yeux, à tel point qu'il nous est désormais impossible d'imaginer notre vie personnelle et professionnelle sans elles. Selon ces auteurs, l'Informatique Ubiquitaire a pris une forme différente de celle attendue par Weiser, dans laquelle les dispositifs mobiles représentent l'élément central de notre vie quotidienne. Ces auteurs soutiennent que nous sommes continuellement en train d'utiliser des ressources informatiques dans notre vie courante sans forcément les percevoir en tant qu'ordinateurs. L'informatique Ubiquitaire est ainsi une réalité sous la forme d'un environnement densément peuplé de ressources informatiques et de communication.

1.1.2. Impact de l'Informatique Pervasive sur les Systèmes d'Information (SI)

La dernière décennie a été remarquablement marquée par le changement dans la manière dont nous travaillons et dans la manière dont nous nous appuyons sur les technologies. Nous passons d'un *modèle statique*, dans lequel les travailleurs n'interagissent avec un processus métier que durant leur « temps de travail » et dans des circonstances bien définies (*e.g.* assignés à leur ordinateur de bureau), à un *modèle dynamique*, dans lequel ils se caractérisent par leur mobilité, permise par l'évolution des réseaux sans fil et des dispositifs mobiles.

Ainsi, les SI sont confrontés à un environnement pour lequel ils n'ont pas été prévus. En effet, l'arrivée de l'Informatique Ubiquitaire, au sein des organisations, a directement impacté les Systèmes d'Information (SI). La mobilité qu'apportent ces nouvelles technologies a étendu les SI bien au-delà des frontières physiques de l'organisation. Ceci revient à dire que l'évolution de ces technologies mobiles et pervasives a ouvert de nouvelles perspectives et a changé le mode d'accès à ces systèmes. Nous assistons au passage graduel d'un paradigme entièrement fondé sur les *desktops* à un paradigme mixte, *intégrant des dispositifs multiples et très hétérogènes* : *desktops*, dispositifs mobiles et ressources intégrées à l'environnement physique (Kourouthanassis et Giaglis, 2006).

Depuis leur apparition, les Systèmes d'Information n'ont cessé d'évoluer et de progresser dans la perspective d'améliorer la productivité et l'efficacité au sein de l'organisation, comme l'illustre la Figure 1. Au début, un Système d'Information (SI) a été conçu comme *une combinaison de pratiques de travail, d'information, d'individus, et de technologies de l'information en vue d'atteindre certains objectifs* (Alter, 1992). Par la suite, un SI a été défini comme un *ensemble organisé de ressources (individus, matériel, logiciel, progiciel, bases de données, procédures) qui permettent d'acquérir, de traiter, de stocker, et de communiquer l'information sous différentes formes au sein d'une organisation* (Reix, 2004). Récemment, Nurcan (Nurcan, 2012) a synthétisé toutes ces définitions en présentant les Systèmes d'Information comme étant *le cœur stratégique de l'entreprise, rassemblant un ensemble organisé de ressources technologiques et humaines visant à (i) aider la réalisation des*

activités de l'organisation, et (ii) faciliter/servir l'accomplissement des objectifs métier fixés pour et par cette organisation.

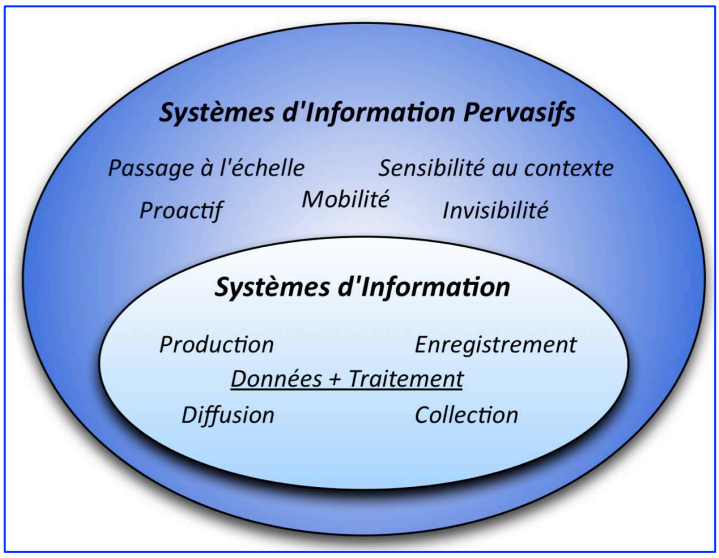


Figure 1. L'émergence des Systèmes d'Information Pervasifs

Les Systèmes d'Information sont devenus un élément clé des organisations. Selon Henderson et Venkatrama (Henderson et Venkatraman, 1993), « *posséder un système d'information efficace et efficient supportant les stratégies métiers et les processus qui y sont rattachés est rapidement devenu un facteur clé de succès* ». Ainsi, le rôle des SI est devenu plus stratégique au sein des organisations : ils contribuent à la mise en place des différents processus métiers propres à ces organisations, dont le succès ou l'échec peut avoir d'importantes conséquences pour leur survie.

Avec le développement des nouvelles technologies, les directeurs des Systèmes d'Information (DSI) sont confrontés à de nouveaux modes d'interaction avec leur SI. L'évolution des SI devient ainsi inévitable. Ces systèmes doivent s'adapter aux nouvelles technologies et aux nouveaux modes d'accès mis à disposition de leurs utilisateurs. Or, cette évolution ne doit pas être subie, mais choisie. Nous assistons donc aujourd'hui à l'émergence d'une nouvelle génération de Systèmes d'Information : les « *Systèmes d'Information Pervasifs* » (SIP). Nous assistons aujourd'hui au passage progressif des technologies de l'information (IT) à l'arrière-plan. En d'autres termes, les Systèmes d'Information sont là aujourd'hui pour surveiller les activités des utilisateurs, pour assembler et traiter les informations et pour intervenir lorsque cela est nécessaire (au lieu d'être uniquement déclenchés et manipulés directement par l'utilisateur) (Kourouthanassis et Giaglis, 2006) .

1.1.3. Les Systèmes d'Information Pervasifs (SIP)

La notion de Système d'Information Pervasif est au cœur de l'évolution mentionnée ci-dessus. Cette nouvelle classe de SI représente l'avenir de ces systèmes. Elle apporte de

nouvelles opportunités, notamment par la prise en compte de l'environnement et la possibilité d'offrir des services innovants.

A l'inverse des SI traditionnels, les Systèmes d'Information Pervasifs s'intègrent progressivement à l'environnement physique (Kourouthanassis et Giaglis, 2006). Cette nouvelle classe de SI se distingue des SI traditionnels par l'environnement pervasif dans lequel ils émergent. Contrairement aux SI traditionnels, intégrés notamment à un environnement *desktop classique* qui limite le mode d'interaction de l'utilisateur (modes d'accès stationnaires), les SIP s'ouvrent à d'autres *dispositifs plus évolués*, offrant à l'utilisateur des espaces plus étendus d'interaction avec le SI : des dispositifs mobiles transportables par l'utilisateur, dispositifs directement intégrés à l'environnement autour. Ainsi, comme le mentionnent Kourouthanassis *et al. (Kourouthanassis et al., 2007)*, en dehors des interactions physiques uniquement avec le système, un SIP peut également intégrer des éléments d'interactions mobiles avec des dispositifs ou des objets de l'environnement physique d'une manière naturelle et discrète.

Par conséquent, à l'opposé des SI traditionnels, complétement maîtrisés et bornés, les futurs SIP visent à s'intégrer à un environnement dynamique et hétérogène. D'une part, selon (Kourouthanassis et Giaglis, 2006), ces SIP s'inscrivent dans un environnement particulièrement dynamique, composé d'une multitude d'artefacts capables de percevoir le contexte de l'utilisateur et de gérer sa mobilité. D'autre part, ces SIP offre, par cette intégration à un environnement hétérogène, une interaction continue avec le SI là où on est et quand on le souhaite. Toujours selon (Kourouthanassis et Giaglis, 2006), contrairement aux SI traditionnels, dont l'intelligence réside dans l'ordinateur *desktop*, les SIP doivent faire résider cette intelligence au-delà de l'ordinateur en l'intégrant dans le monde physique. De plus, les SIP doivent être proactifs, réagissant aux stimuli de l'environnement, à l'encontre des SI traditionnels, dans lesquels une réponse du système est forcément précédée d'une action de l'utilisateur.

Cette nouvelle génération des SI, avec les nouvelles opportunités qu'elle apporte, représente l'avenir de ces systèmes. Mais il est à noter que cette nouvelle génération apporte également son lot de défis : hétérogénéité, dynamisme, accès ubiquitaire aux SI, etc. À ce jour, les DSI sont livrés à eux-mêmes. Ils ne disposent d'aucun moyen leur permettant de concevoir, de construire et de mieux comprendre cette nouvelle classe de SI.

1.2. PROBLEMATIQUE

A travers ces *nouvelles technologies, de nouvelles opportunités de services peuvent s'offrir aux utilisateurs*. Selon (Kourouthanassis et Giaglis, 2006), les SIP se caractériseraient non seulement par l'hétérogénéité des dispositifs impliqués, mais également par l'interaction continue rendue possible par ces dispositifs, mobiles ou intégrés à l'environnement physique, ce qui ouvre *la possibilité d'offrir aux utilisateurs de nouveaux services innovants.*

Toutefois, afin de concevoir de tels SI évoluant dans un environnement pervasif, il est nécessaire de bien comprendre les caractéristiques et les exigences auxquelles ces nouveaux systèmes seraient soumis. Bien évidemment, les caractéristiques principales des SIP sont l'hétérogénéité des ressources, des infrastructures, des terminaux. Cette *hétérogénéité* vient justement de l'environnement pervasif dans lequel ces systèmes sont intégrés. Cette hétérogénéité se retrouve également au niveau des services offerts par le SI, avec de multiples technologies possibles (services Web traditionnels, composants OSGI, pour ne citer qu'eux). Les utilisateurs s'orientent ainsi vers un monde de plus en plus hétérogène avec des données et des services répartis à différents niveaux.

Il est illusoire d'imaginer que cette *hétérogénéité* disparaîtra avec le temps. Ces *environnements resteront toujours complexes et extrêmement denses d'un point de vue technologique.* Les SIP doivent gérer cette hétérogénéité et veiller à la bonne interaction entre l'utilisateur et l'environnement physique (Kourouthanassis et Giaglis, 2006). Se pose alors la question : comment l'utilisateur peut-il faire face à cette hétérogénéité et se concentrer sur son propre objectif et non sur la technologie elle-même ? Effectivement, il faut masquer cette hétérogénéité et cacher la complexité de l'environnement. L'élément clé devient ainsi la gestion de la *transparence*. Or il nous paraît impossible d'homogénéiser toutes les technologies existantes, même dans un cadre plus contrôlé comme celui des SI. Par ailleurs, *les offres de services risquent d'augmenter significativement*. Selon Savidis (Savidis, 2010), dans un environnement pervasif, nous sommes confrontés à une évolution rapide du spectre et du nombre de services disponibles pour tout type d'utilisateur, à n'importe quel endroit et à tout moment. Dans la vie quotidienne, le temps moyen passé dans l'utilisation de ces services varie d'une dizaine de secondes à quelques minutes, alors qu'ils sont impliqués dans un nombre croissant d'activités quotidiennes, hebdomadaires ou mensuelles (Savidis, 2010). En effet, il y a une claire tendance à développer et à offrir de plus en plus de services et notamment de services personnalisés, comme le mentionne Wolisz (Wolisz, 2010).

Les Systèmes d'Information sont ainsi confrontés à un environnement pour lequel ils n'ont pas été particulièrement conçus (hétérogénéité, offre de services grandissante, etc.). Les Systèmes d'Information Pervasifs se veulent alors une réponse à cette importante évolution des SI. Par contre, *ils se doivent de gérer l'hétérogénéité de l'environnement et l'offre de services de manière transparente afin de rassurer les utilisateurs*.

Or les concepteurs des SIP se trouvent, aujourd'hui, démunis face à une notion relativement nouvelle. Effectivement, les SIP constituent une nouvelle génération des SI qui est difficile à conceptualiser, avec peu de formalismes disponibles. Autant dire qu'ils n'ont rien à leur disposition pour les aider à concevoir de tels systèmes. Partant de ce fait, il est nécessaire de mettre en place un cadre plus formel permettant d'aider les concepteurs à mieux comprendre les SIP et surtout à mieux les maîtriser, tout en assurant la transparence nécessaire à ces systèmes.

Par ailleurs, force est de constater que l'environnement pervasif est un environnement hautement *dynamique* qui varie en fonction de l'utilisateur (ses actions, sa mobilités, etc.) et

de ses éléments. Ceci ajoute aux SIP une autre caractéristique, au-delà de l'hétérogénéité : le *dynamisme*. Selon (Hagras, 2011), la nature dynamique des environnements pervasifs impose une capacité d'adaptation à des conditions d'opération changeantes et à des utilisateurs dont les préférences et le comportement sont également variables. Un système pervasif doit être capable d'accomplir les fonctionnalités sollicitées, malgré les changements dans les conditions environnantes ou dans l'état du système (Römer et Friedemann, 2010). En d'autres termes, les SIP doivent *s'adapter aux changements* de l'environnement afin de gérer son dynamisme. Ainsi, afin d'assurer la transparence nécessaire, les SIP doivent être *sensibles au contexte*, permettant la prise en compte de l'environnement pervasif.

Selon Baldauf et al. (Baldauf et al., 2007), les *systèmes sensibles au contexte* se caractérisent par leur capacité à adapter leur fonctionnement, par la prise en compte du contexte environnant, afin d'augmenter leur utilisabilité et leur efficacité. Dans ce cadre, la notion de contexte représente « *l'ensemble des caractéristiques de l'environnement physique ou virtuel qui affecte le comportement d'une application et dont la représentation et l'acquisition sont essentielles à l'adaptation des informations et des services* » (Gensel et al., 2008). Pour (Kourouthanassis et al., 2008), la *sensibilité au contexte* représente la capacité d'un système à percevoir les informations contextuelles relatives à l'utilisateur, au système lui-même et à l'environnement afin de pouvoir adapter ses fonctionnalités de manière dynamique et proactive, réagissant aux stimuli de l'environnement.

Cependant, cette sensibilité au contexte ne doit pas se faire au dépend de la transparence. Selon Dey (Dey, 2011), lorsque les utilisateurs ont des difficultés à former un modèle mental de l'application, ils ont moins envie de l'adopter et de l'utiliser. En plus de leurs capacités d'adaptation au contexte et d'un comportement proactif, les SIP doivent rester *compréhensibles* à leurs utilisateurs, d'autant plus qu'il s'agit, avant tout, de Systèmes d'Information. Les SI sont là pour répondre aux besoins des utilisateurs. Il est primordial de rassurer les utilisateurs afin qu'ils gardent leur confiance sur ces systèmes. De ce fait, il faut avoir une vision globale du fonctionnement des SI afin d'assurer un tel niveau de compréhension. Ainsi, il est nécessaire de représenter ce fonctionnement global du système à travers une modélisation de haut niveau. Selon (Hagras, 2011), ces modèles doivent être eux-mêmes transparents et d'interprétation facile aux utilisateurs afin que ceux-ci puissent mieux analyser le système et ses performances.

Partant de ces faits, les SIP se doivent d'être *sensibles au contexte* et *compréhensibles* à leurs utilisateurs afin d'assurer la *transparence* nécessaire et de gérer l'*hétérogénéité*, sans pour autant perdre complétement le caractère *maîtrisé* et *prédictible* propre aux SI. En tant que Systèmes d'Information, les SIP doivent être conçus afin de mieux *satisfaire les besoins* de leurs utilisateurs en prenant en considération leur environnement pervasif. De plus, au contraire des SI traditionnels, les SIP doivent désormais s'adapter à l'environnement et au contexte de l'utilisateur afin de lui offrir le service le plus approprié. Cependant, cette adaptation doit respecter certains critères et ne doit pas se faire n'importe comment et à n'importe quel prix. Effectivement, le comportement d'un SIP, même s'il doit tirer profit de l'environnement dynamique et des opportunités qu'un tel environnement peut lui offrir, se

doit de rester *prédictible*, afin d'assurer la gouvernance de ces systèmes et la confiance des utilisateurs en eux. Le développement d'un SIP répondant à ces besoins est aujourd'hui un problème encore non résolu. Il n'existe pas à l'heure actuelle de modèle ou de méthode permettant aux concepteurs et aux développeurs de prendre en compte ces besoins lors de la conception d'un SIP.

D'une part, les recherches au niveau des systèmes pervasifs se sont essentiellement concentrées sur le niveau technique. Des efforts importants ont été consacrés sur l'adaptation au contexte (Baldauf et al., 2007) (Chaari et al., 2008b) (Preuveneers et al., 2009), surtout à la localisation (Coronato et al., 2009) (Varshavsky et Patel, 2009) et aux terminaux (Lemlouma, 2004) (Yang et Shao, 2007). Nous constatons aujourd'hui les limitations de ces approches qui ne tiennent pas compte des exigences derrière l'expérience de l'utilisateur. Plusieurs possibilités peuvent être offertes à l'utilisateur, qui n'est toujours pas capable de comprendre ce qui lui a été proposé, ce qui nuit à la transparence d'utilisation de ces systèmes. Pour nous, *la clé du succès serait donc d'assurer un certain niveau de compréhension à ces systèmes pervasifs sensibles au contexte*.

D'autre part, les recherches autour de la conception des SI ont été nombreuses, et nombreux sont les chercheurs à souligner l'importance de la notion d'intention (Santos et al., 2009)(Rolland et al., 2010)(Deneckere et Kornyshova, 2010). Une *intention* représente ce qu'attend l'utilisateur de l'exécution du service. Elle représente donc la vision de l'utilisateur sur les fonctionnalités qu'il désire dans un service (Fensel et al., 2011). Cette vision intentionnelle place la notion de service à un niveau d'abstraction plus élevé : le service est là afin de conduire son utilisateur à la satisfaction d'une intention. Il s'agit, selon (Rolland et al., 2010), de combler le fossé qui sépare une vision purement technique d'une vision purement métier des services, centrée sur l'utilisateur et ses besoins. La notion d'intention place donc le service à un niveau plus proche de celui de l'utilisateur final : quelle que soit la technologie utilisée, le service est défini pour satisfaire un besoin, un but exprimé par l'utilisateur.

La notion d'intention a été souvent reliée à la notion de service. Plusieurs travaux ont, en effet, considéré la notion de service sous un angle intentionnel (Rolland et al., 2010) (Mirbel et Crescenzo, 2010a) (Santos et al., 2009) (Fensel et al., 2011). Ces approches considèrent qu'un utilisateur cherche, avant tout, à satisfaire une intention, et que les services constituent seulement un moyen de l'atteindre. Mais on constate que cette notion n'a été confrontée à la notion de contexte que rarement (Mirbel et Crescenzo, 2010a) (Santos et al., 2009) (Deneckere et Kornyshova, 2010). En effet, même si la notion d'intention permet au système de mieux comprendre les besoins réels des utilisateurs, l'influence d'un environnement pervasif dans l'émergence et la satisfaction des intentions reste encore peu explorée.

Ainsi, nous pouvons résumer notre problématique à un *problème de conception et de réalisation d'un SIP qui répond à tous les besoins de transparence, d'adaptation à l'environnement et d'adaptation à l'utilisateur d'un SIP*. En effet, avec le manque de modèle et de méthode permettant de prendre en compte tous ces besoins, la DSI (concepteur du SIP) se trouve face à de grandes difficultés rendant difficiles la conception et la réalisation d'un

SIP transparent et centré utilisateur. Elle se trouve démunie face à une nouvelle génération de SI qui jusqu'à présent n'a pas été véritablement mise en place avec des formalismes appropriés dans l'objectif d'aider et d'orienter les concepteurs de ce système.

En somme, nous pensons que la conception et la réalisation d'un SIP doit impérativement répondre aux problèmes suivants :

- ***Transparence*** : contrairement aux SI traditionnels, les SIP doivent gérer l'hétérogénéité des environnements et des services, et ils doivent le faire de manière transparente à l'utilisateur. Or, à ce jour, peu d'outils, de formalismes et de méthodes destinés aux SIP sont à dispositions de leurs concepteurs.
- ***Adaptation à l'environnement*** : contrairement aux SI traditionnels, les SIP doivent être conçus pour opérer dans un environnement pervasif. Ils doivent s'adapter au caractère dynamique de ces environnements, sans pour autant perdre la maitrise propre aux SI. L'équation est délicate et, à nouveau, peu d'outils ou de formalismes s'offrent aux concepteurs de ces systèmes.
- ***Adaptation aux utilisateurs*** : les SIP doivent être conçus de manière à s'adapter non seulement à leur environnement, mais également à leurs utilisateurs. Ceux-ci ont des besoins auxquels les SIP doivent répondre de la manière la plus adaptée possible, tout en gardant la transparence nécessaire pour que le système disparaisse derrière la satisfaction des besoins.

1.3. HYPOTHESES

Afin de traiter les problèmes soulevés dans la section précédente, nous considérons les hypothèses de travail suivantes :

- **Hypothèse principale** : Une approche centrée utilisateur permettrait de garantir un environnement pervasif transparent au sein des SIP.

  Le développement d'une couche de haut niveau centrée utilisateur permettrait aux SIP de cacher la complexité de l'environnement pervasif qui se caractérise par son hétérogénéité et son dynamisme. Ceci permettrait alors de garantir la transparence de cet environnement pour l'utilisateur final.

  Nous croyons qu'une telle approche doit se concentrer à la fois sur l'adaptation (la prise en compte de l'environnement afin que celui-ci puisse devenir plus transparent à l'utilisateur) et sur les besoins de l'utilisateur (considérer aussi bien la raison pour laquelle il sollicite une action que la manière dont elle est réalisée).

Les hypothèses ci-dessous détaillent cette vision.

- **Hypothèse 1** : La prise en compte de l'intention permettrait de mieux répondre aux besoins de l'utilisateur dans un Système d'Information Pervasif

Dans un Système d'Information, le but principal est de répondre aux besoins de l'utilisateur. L'expression de ces besoins sous la forme d'intention permettrait aux SIP de comprendre le 'pourquoi' d'une action, de mieux assimiler ce que l'utilisateur cherche réellement, et de répondre au mieux à ces besoins en proposant le service le plus approprié.

- **Hypothèse 2 :** La sensibilité au contexte permettrait de mieux gérer le dynamisme de l'environnement pervasif dans les SIP

  Dans un environnement pervasif, la sensibilité au contexte joue un rôle central. Dans le cadre d'un SIP, elle permettrait de mieux gérer le dynamisme de l'environnement en percevant les informations contextuelles relatives à l'utilisateur et à l'environnement afin de pouvoir mieux adapter ses fonctionnalités de manière dynamique et proactive.

- **Hypothèse 3 :** L'intention de l'utilisateur émerge dans un contexte donné

  L'intention n'est pas le fruit du hasard. Elle représente le besoin d'un utilisateur. Or ce besoin émerge dans un contexte donné. En d'autres termes, la notion d'intention est directement liée à la notion de contexte. Nous pensons qu'une intention n'a de sens que lorsqu'on la considère dans un contexte donné.

- **Hypothèse 4** : La réalisation de l'intention est valide dans un contexte, le contexte influence le choix de la réalisation

  Le contexte dans lequel émerge une intention peut influencer considérablement la manière dont cette intention peut être satisfaite, et donc influencer sa réalisation.

- **Hypothèse 5** : La prise en compte de l'intention et du contexte permettrait d'assurer un Système d'Information Pervasif transparent et compréhensible à l'utilisateur

  Un Système d'Information Pervasif doit être considéré comme étant un SI qui évolue dans un environnement pervasif. La prise en compte de leurs besoins respectifs en termes d'intentionnalité et de sensibilité au contexte permettrait de répondre au mieux aux besoins de l'utilisateur en assurant la transparence nécessaire pour un SIP.

- **Hypothèse 6** : Un mécanisme de prédiction de services, capable d'anticiper les besoins de l'utilisateur, pourra améliorer la transparence générale du système

  Le développement d'un mécanisme de prédiction de services qui permet d'anticiper les besoins des utilisateurs et de répondre à leurs intentions futures dans un contexte donné, permettrait aux SIP de cacher la complexité de l'environnement pervasif et d'améliorer la transparence de cet environnement pour l'utilisateur final.

Toutes ces hypothèses représentent le fondement de notre vision intentionnelle et contextuelle des Système d'Information Pervasif. Cette vision est présentée dans la section suivante et sera mise en place en présentant un cadre conceptuel des SIP, une démarche de conception d'un SIP et une architecture de gestionnaire de SIP conforme au cadre conceptuel.

1.4. APERÇU DE LA PROPOSITION

Afin de répondre à notre problématique, nous proposons notre vision des SIP. Cette vision se base sur l'orientation *service*, la *sensibilité au contexte* et sur une *approche intentionnelle* afin de résoudre les problèmes de *transparence*, d'*adaptation à l'environnement* et d'*adaptation aux utilisateurs* que nous avons soulevés ci-dessus.

Notre vision vise à concevoir un SIP transparent qui (i) gère l'hétérogénéité et la dynamique de l'environnement pervasif ; (ii) assure un certain niveau de contrôle et de maitrise nécessaire dans le cadre d'un SI ; et (iii) comprend les exigences et les besoins réels des utilisateurs derrière leur demande d'un service donné.

Les hypothèses énumérées ci-dessus représentent le fondement de cette vision des SIP basée sur les notions d'*intention*, de *contexte* et de *services*. Ceci représente une vision *centrée utilisateur des SIP*, permettant de gérer l'hétérogénéité et le dynamisme de l'environnement à travers une approche intentionnelle et contextuelle. En effet, dans la perspective d'assurer la transparence et la compréhension nécessaire pour la conception d'un SIP, nous considérons les SIP et leurs éléments à la fois sous l'angle des SI et celui des environnements pervasifs, en observant leurs besoins respectifs de contrôle, d'intentionnalité et de sensibilité au contexte.

Cette vision est orientée *services,* car elle permet de répondre au besoin de *gestion de l'hétérogénéité technique* de l'environnement dans lequel évoluent les SIP et des actions que le système propose afin de satisfaire les besoins des utilisateurs. Ce choix repose sur la caractéristique principale des services, à savoir leur i*ndépendance par rapport aux aspects technologiques et à leur implémentation*, ce qui nous permet ainsi de masquer l'hétérogénéité technologique des environnements pervasifs.

De plus, notre vision est orientée *contexte*, car elle permet d'adapter les SIP au contexte de l'utilisateur et à l'environnement. Elle permet également de mieux gérer l'hétérogénéité et le dynamisme de l'environnement pervasif.

Enfin, notre vision est orientée *intention*, permettant de répondre d'une façon personnalisée aux besoins de l'utilisateur. Cette notion d'intention formalise les besoins de l'utilisateur. Nous pensons que, dans le cadre des SIP, cette notion est nécessaire pour que ces systèmes comprennent mieux l'utilisateur et répondent à son besoin de la manière la plus appropriée.

Par ailleurs, nous mettons l'accent, dans le cadre de notre vision centrée utilisateur des SIP, sur l'étroite relation entre les notions d'intention, de contexte et de service. Comme l'illustre

nos hypothèses, l'intention de l'utilisateur émerge dans un contexte donné. De plus, les réalisations de ses intentions ne sont valides que dans un contexte d'utilisation bien défini. Dans ce cadre, la notion de contexte représente un élément important dans le processus d'adaptation d'un système à l'utilisateur, auquel nous souhaitons ajouter la notion d'intention.

En se basant sur ces notions d'intention, de contexte et de service et en exploitant la relation qui les lie, nous proposons une nouvelle vision centrée utilisateur d'un SIP transparent, non intrusif et compréhensible à l'utilisateur.

Par la suite, nous proposons une solution plus globale pour concrétiser notre vision intentionnelle et contextuelle des SIP orientés services. Nous proposons une solution pour aider la DSI à concevoir un SIP en présentant un *cadre conceptuel des SIP* (*cf.* Chapitre 5) décrivant et formalisant l'ensemble de ses éléments. Nous proposons ensuite des mécanismes de *découverte* (*cf.* Chapitre 7) et de *prédiction* (*cf.* Chapitre 8) de services qui sont intégrés dans une *architecture de gestionnaire de SIP* (*cf.* Chapitre 9) qui est conforme à ce cadre conceptuel. Finalement, nous proposons une *démarche méthodologique* (*cf.* Chapitre 10) de conception et de réalisation d'un SIP qui supporte le passage entre le cadre conceptuel et l'implémentation de l'architecture proposée.

Notre vision centrée utilisateur est déclinée sur quatre dimensions :

- ***Dimension conceptuelle***, à travers un *cadre conceptuel* dans la perspective d'aider la Direction des Systèmes d'Information (DSI) à mieux conceptualiser de tels systèmes et ses éléments (*i.e.* le service qu'ils offrent et les éléments de contexte observés) ;
- ***Dimension fonctionnelle***, grâce aux mécanismes de *découverte de services* et de *prédiction* de services en utilisant l'approche intentionnelle et sensible au contexte des SIP proposée ;
- ***Dimension système***, par l'*architecture de gestionnaire de SIP* en mettant en œuvre la vision intentionnelle et contextuelle conforme au cadre conceptuel. Elle intègre également des mécanismes de découverte et de prédiction de services ;
- ***Dimension support***, avec la *démarche méthodologique de conception* guidant le design des SIP du cadre conceptuel jusqu'à la description des services au dessus de l'architecture du système.

Notre vision centrée utilisateurs des SIP, ainsi que sa concrétisation selon les quatre dimensions citées ci-dessus, seront présentées plus en détails dans le Chapitre 4.

1.5. ORGANISATION DE LA THESE

Ce travail est organisé comme suit :

- Le *deuxième chapitre*, représente un *état de l'art* sur les *Systèmes d'Information Pervasifs* et sur *la notion de contexte.* Nous évoquons, dans ce chapitre, les principaux

thèmes liés à notre travail, à savoir la *sensibilité au contexte* et les *Systèmes d'Information Pervasifs* ;

- Le *troisième chapitre*, représente un *état de l'art* sur *l'orientation service*. Dans ce chapitre, nous évoquons les principaux thèmes liés à notre travail, à savoir les *systèmes orientés services* et les différentes tendances existantes, les systèmes intentionnels, etc. Par la suite, nous présentons les challenges pour les Systèmes d'Information Pervasifs orientés services, plus spécifiquement la découverte et la prédiction dynamique de services ;
- Le *quatrième chapitre*, résume notre *problématique* et présente plus en détail notre *proposition et la solution globale* ;
- Le *cinquième chapitre*, présente le cadre conceptuel des SIP, qu'on a appelé « *espace de services ».* Dans ce chapitre, nous présentons une conceptualisation des différents éléments constituant l'espace de services, à savoir les *services* et les *capteurs* ;
- Le *sixième chapitre*, détaille la *description sémantique des services*. Ce chapitre explique notre extension de OWL-S pour inclure les informations intentionnelles et contextuelles des services en conformité avec l'espace de services ;
- Le *septième chapitre*, présente le *mécanisme de découverte de services guidé par le contexte et l'intention.* Dans ce chapitre nous présentons notre algorithme de découverte dynamique des services intentionnels et contextuels, ainsi que notre implémentation et évaluation de cet algorithme ;
- Le *huitième chapitre*, illustre le *mécanisme de prédiction de services basé sur l'intention et le contexte*. Dans ce chapitre nous présentons notre processus d'apprentissage et de prédiction dynamique des services intentionnels et contextuels, ainsi que notre implémentation et évaluation de ce processus ;
- Le *neuvième chapitre*, illustre notre architecture de gestionnaire de SIP et présente ses différents composants et les interactions entre eux, afin de mettre en œuvre les différents concepts de l'espace de services ;
- Le *dixième chapitre*, expose notre démarche méthodologique supportant le passage du cadre conceptuel des SIP vers l'architecture de gestionnaires des SIP. Nous illustrons, par la suite, cette méthodologie à travers un cas d'étude ;
- Finalement, nous finalisons ce manuscrit par le *onzième chapitre* qui conclut ce travail de recherche et ouvre de nouvelles perspectives.

Chapitre 2. Systemes d'Information Pervasifs et la notion de contexte

2.1. Introduction

L'émergence des nouvelles technologies et la démocratisation des terminaux mobiles ont impacté remarquablement notre quotidien et notre façon d'utiliser les systèmes et les services disponibles. Ceci représente l'apparition de l'Informatique Pervasive, laquelle vise à intégrer les technologies d'une façon invisible dans notre vie, rendant les services offerts par les systèmes disponibles n'importe où et n'importe quand. Cette arrivée de l'Informatique Pervasive impacte sérieusement les Systèmes d'Information (SI) au sein des organisations.

Au cœur de l'Informatique Pervasive se trouve la notion de sensibilité au contexte. La notion de contexte a été largement étudiée et discutée dans ce domaine. Elle représente un concept très vague qui fait apparaître diverses définitions selon différents points de vue allant des plus généraux, applicables à tous les domaines, au plus spécifiques s'appliquant à un type d'application ou domaines précis. Cette variété de définition a fait émerger de multiples modèles dans la perspective de représenter et de limiter la notion de contexte. Ces modèles de contexte ont fait l'objet de plusieurs années de recherche qui ont conduit à une modélisation sémantique plus compréhensible et plus significative, faisant appel aux ontologies et à de techniques de raisonnement plus puissantes. Ces modèles de contexte sont utilisés dans les processus d'adaptation et de personnalisation au sein des Systèmes Sensibles au Contexte afin d'offrir les services les plus appropriés et les mieux adaptés à l'utilisateur.

Ce chapitre présente un état de l'art sur la notion de contexte et sur les Systèmes d'Information Pervasifs (SIP). Nous commençons par illustrer l'émergence et l'historique du domaine de l'Informatique Pervasive et des Systèmes Sensibles au Contexte. Nous présentons, par la suite, la notion de contexte, nécessaire à ces systèmes, exposée sous différentes définitions, caractéristiques et dimensions. Dans cette partie, nous attribuons une attention particulière aux différentes modélisations de contexte illustrées à la littérature et au processus de gestion de contexte qui représentent le fondement des Systèmes Sensibles au Contexte. De plus, face à la multitude de modèles de contexte, nous proposons, à la fin de cette partie, un cadre d'analyse et de comparaison des modèles existants. Finalement, nous présentons l'impact de l'émergence de l'Informatique Pervasive sur les SI faisant apparaître une nouvelle classe de SI appelée les *Systèmes d'Information Pervasifs* (SIP).

2.2. L'Informatique Pervasive et les Systemes Sensibles au Contexte

Dans cette section, nous présentons l'historique de l'apparition du domaine de l'Informatique Pervasive et des Systèmes d'Information Pervasifs.

2.2.1. Définition et historique du domaine de l'Informatique Pervasive

Depuis plusieurs années, nous sommes témoins d'une importante évolution des nouvelles technologies (*Smartphones*, *tablettes*, *3G,* etc.) et de la façon dont nous les utilisons pour accéder aux différents systèmes et services qui prennent place dans notre quotidien. Ceci représente l'ère de l'Informatique Pervasive, dorénavant invisible à nos yeux, faisant partie intégrante de notre quotidien, à tel point qu'il nous est désormais impossible d'imaginer notre vie personnelle et professionnelle sans elle. Cette nouvelle ère vise à intégrer discrètement les ordinateurs dans la vie de tous les jours des utilisateurs à la maison et au travail (Che et al. 2007). L'objectif est de rendre plus confortable la vie au quotidien des utilisateurs. Elle a tendance vers la miniaturisation des dispositifs électroniques et leur intégration à n'importe quel objet du quotidien, favorisant ainsi l'accès aux informations dont nous avons besoin partout et à tout moment. Ainsi, l'Informatique Pervasive émerge comme un nouveau paradigme fournissant des services informatiques à tout moment et n'importe où.

Cette vision d'une Informatique invisible et intégrée à l'environnement a été soutenue depuis plusieurs années par l'Informatique Pervasive (ou Ubiquitaire) (Weiser 1991). Selon plusieurs observations, Weiser a constaté que le ratio d'ordinateur par personne n'a cessé d'augmenter considérablement. Dans son article (Weiser 1991), il soutient que « *Les technologies les plus profondément enracinées sont les technologies invisibles. Elles s'intègrent dans la trame de la vie quotidienne jusqu'à en devenir indiscernables* ». Cette vision des technologies présentes mais non intrusives, qui accompagnent les usages du quotidien et les déplacements des utilisateurs, a guidé vingt années de recherche en Informatique Pervasive (Bell et Dourish, 2007).

L'Informatique Pervasive a été envisagée par Weiser (Weiser, 1991) comme la troisième ère de l'informatique moderne, dans laquelle l'utilisateur est confronté à une multitude d'ordinateurs sans fil communiquant entre eux discrètement. En effet, un changement radical dans l'informatique a été souligné par Weiser (Weiser, 1991) : partant de l'époque ou plusieurs utilisateurs partagent le même ordinateur (ère de l'ordinateur central « *mainframe* »), vers l'époque de la relation personnelle avec l'ordinateur (ère de l'ordinateur personnel), en arrivant à l'émergence d'un monde où l'utilisateur interagit en toute transparence avec une multitude d'ordinateurs. L'ère de l'Informatique Pervasive représente l'époque actuelle qui est caractérisée par l'évolution des technologies mobiles et par la démocratisation des dispositifs et des réseaux mobiles. Ceci a entraîné progressivement la croissance remarquable des ordinateurs intégrés dans la vie quotidienne.

Selon Satyanarayana (Satyanarayana 2001), l'Informatique Pervasive englobe l'informatique mobile, mais elle va beaucoup plus loin. L'informatique mobile s'appuie sur les fondations de systèmes distribués, des réseaux mobiles et des systèmes sensibles à l'énergie. Elle englobe tout ceci en y intégrant quatre axes de recherche complémentaires : les *espaces intelligents*, l'*invisibilité*, le *passage à l'échelle localisée* et le *conditionnement inégal* (Satyanarayana 2001). Le premier axe, les *espaces intelligents*, est l'utilisation efficace des

espaces intelligents qui embarquent le monde physique dans le monde conceptuel. Le deuxième axe, l'*invisibilité*, renvoie à ce que Weiser décrit comme *une intégration transparente de l'informatique dans le tissu de notre vie quotidienne*. Ceci signifie, comme l'a mentionné Weiser (Weiser 1991), la disparition de la technologie dans la conscience de l'utilisateur. Le troisième axe, le *passage à l'échelle localisée,* comprend l'emploi de méthodes qui réduisent l'intensité des interactions de l'utilisateur avec les services offerts par le système (basées sur la distance entre eux). Finalement, le quatrième axe, le *conditionnement inégal*, fait référence aux techniques permettant de masquer le conditionnement intégral de l'environnement. Ceci nécessite un espace de transition, où les applications informatiques traditionnelles et les infrastructures sont peu à peu remplacées par d'autres, présentant les caractéristiques de l'Informatique Pervasive.

Toutefois, une certaine confusion persiste quant à savoir si cette vision est une réalité ou un avenir proche. Bell et Dourish (Bell et Dourish, 2007) avancent l'idée selon laquelle l'Informatique Pervasive est déjà une réalité. Pour ces auteurs, elle fait partie intégrante de notre quotidien, notamment à travers les nouvelles technologies. Celles-ci sont effectivement devenues presque invisibles à nos yeux, à tel point qu'il nous est désormais impossible d'imaginer notre vie personnelle et professionnelle sans ces technologies. Selon Bell et Dourish (Bell et Dourish, 2007), l'Informatique Pervasive a pris une forme différente de celle attendue par Weiser, à travers les dispositifs mobiles centraux à notre vie quotidienne. Ces auteurs soutiennent que nous sommes continuellement en train d'utiliser des ressources informatiques dans notre vie courante sans forcément les percevoir en tant qu'ordinateurs. L'Informatique Pervasive est ainsi une réalité sous la forme d'un environnement densément peuplé de ressources informatiques et de communication (Bell et Dourish, 2007). Ce point de vue est partagé par Greenfield (Greenfield, 2006) selon qui l'Informatique Pervasive représente une informatique sans ordinateur, dans laquelle le traitement de l'information est omniprésent dans la vie quotidienne.

De part l'intégration des différents dispositifs à l'environnement et à notre vie quotidienne, on peut caractériser l'Informatique Pervasive par son *hétérogénéité*, hétérogénéité des ressources et de l'infrastructure. Il est illusoire d'imaginer que cette hétérogénéité disparaîtra avec le temps. Ces environnements resteront toujours complexes et extrêmement denses d'un point de vue technologique. Au-delà de cette hétérogénéité, les environnements pervasifs se caractérisent également par leur dynamique. Selon (Hagras, 2011), la nature dynamique de ces environnements leur impose une capacité d'adaptation à des conditions d'opération changeantes et à des utilisateurs dont les préférences et le comportement sont également variables. L'informatique Pervasive implique donc la *sensibilité au contexte*.

2.2.2. Systèmes Sensibles au Contexte

Les années quatre-vingt-dix ont été caractérisées par l'émergence des *Systèmes Sensible au Contexte.* Cette nouvelle notion est apparue à travers notamment les travaux de Schilit et Theimer (Schilit et Theimer, 1994), Schilit et al. (Schilit et al., 1994), Hull (Hull et al., 1997) et Dey (Dey, 2000). Schilit et Theimer (Schilit et Theimer, 1994) définissent la sensibilité au

contexte comme la *capacité d'une application à découvrir et à réagir aux modifications dans l'environnement où se trouve l'utilisateur*. Selon Schilit et al. (Schilit et al., 1994), les Systèmes Sensibles au Contexte sont définis comme des systèmes qui s'adaptent à la localisation de l'utilisateur et à l'ensemble des personnes, des machines et des dispositifs proches ou accessibles, ainsi qu'aux changements dans le temps de ces éléments. Quelques années plus tard, d'autres définitions de la sensibilité au contexte sont apparues avec les travaux de (Hull et al., 1997) et (Dey, 2000), pour ne citer qu'eux. Hull et al. (Hull et al., 1997) définissent la *sensibilité au contexte* comme la capacité des dispositifs informatiques à détecter, capturer, interpréter et répondre aux aspects de l'environnement de l'utilisateur et aux dispositifs informatiques. Dey (Dey, 2000), quant à lui, présente une définition plus générale. Il considère qu'un *système est sensible au contexte s'il utilise le contexte pour fournir à l'utilisateur des informations ou des services pertinents, où la pertinence dépend de la tâche exécutée par l'utilisateur* (Dey, 2000).

Ainsi, l'une des prémisses essentielles des systèmes sensibles au contexte est d'être conscient des circonstances dans lesquelles se trouve l'utilisateur, d'être capable d'interpréter et de réagir à toute interaction conformément à ces circonstances (O'Hare et O'Grady, 2002). Dans ce cadre, le *contexte* est un élément clé, car il est au centre des mécanismes d'adaptation prônés par ces systèmes dits *sensibles au contexte*. Ces systèmes se caractérisent, en effet, par leur capacité à adapter leur fonctionnement afin d'augmenter leur utilisabilité et leur efficacité, par la prise en compte du contexte environnant (Baldauf et al., 2007). En d'autres termes, un Système Sensible au Contexte est un système qui supporte une certaine variabilité, le choix de la variante dépendant du contexte qui entoure l'exécution du système et son interaction avec les utilisateurs. Le contexte agit ainsi comme un élément extérieur au système qui influence sa variabilité intérieure, une sorte de contrainte qui guiderait le choix de la variante la plus appropriée et le processus d'adaptation la concernant (Najar et al., 2009).

La sensibilité au contexte est une des propriétés les plus importantes d'un système pervasif caractérisant la capacité d'un système à prendre en compte l'environnement, à acquérir des informations sur ce dernier, puis à réagir en conséquence. Il s'agit d'un des piliers pour construire des systèmes mobiles et pervasifs modernes (Schilit et al., 1994) (Dey et Abowd, 2000) (Jones et Grandhi, 2005) (Bolchini et al., 2009). La mobilité de l'utilisateur apportée par l'évolution des nouvelles technologies représente un aspect important conduisant à l'apparition de ces Systèmes Sensibles au Contexte. Selon Dourish (Dourish, 2004), lorsque l'informatique est sortie de l'environnement du bureau (fixe) traditionnel, il est devenu important de suivre à la trace la situation dans laquelle la technologie est utilisée. Par exemple, les utilisateurs mobiles ont besoin d'un contenu informatif qui convient à leur utilisation sous des conditions changeantes (temporelles, spatiales, matérielles, physiques et environnementales) (Carrillo-Ramos et al., 2009). Ces besoins d'adaptation ont guidé la proposition des Systèmes Sensibles au Contexte. Dans ces systèmes, le contexte guide les mécanismes d'adaptation utilisés pour personnaliser le contenu et les services en conséquence.

Même si des solutions sensibles au contexte sont apparues dans différents domaines de recherche, la sensibilité au contexte atteint son utilité maximale lorsqu'elle est appliquée à des

systèmes pervasifs mobiles. La sensibilité au contexte permet également aux services de s'adapter dynamiquement et efficacement à la fois : (i) à la situation actuelle, tels que le lieu physique actuel et/ou l'activité sociale ; et (ii) aux conditions complexes et variables de déploiement typiques des environnements mobiles (rareté des ressources, la connectivité, etc.). Cette capacité à rassembler et à livrer à un service toutes les informations pertinentes pouvant caractériser son environnement d'exécution (les ressources/capacités informatiques, l'emplacement du dispositif physique, les préférences des utilisateurs, etc.) est ainsi devenue une fonction fondamentale pour le développement de systèmes mobiles modernes.

Finalement, il est important de souligner que la notion de *contexte* est considérée comme l'*élément clé* des Systèmes Sensibles au Contexte et, ainsi des Systèmes d'Information Pervasifs. En effet, le contexte est au centre des mécanismes d'adaptation de ces systèmes. Nous discutons, dans la section suivante, les différentes facettes de la notion de contexte.

2.3. LE CONTEXTE

Dans cette section, nous allons procéder à l'étude de la notion de contexte. Nous commençons par présenter, dans la section 2.3.1, les différentes définitions, caractéristiques et dimensions, et nous détaillons, par la suite, les différentes modélisations dans la section 2.3.2. La section 2.3.3 englobe une présentation de la gestion de contexte nécessaire aux processus d'adaptation des Systèmes Sensibles au Contexte. Nous conclurons cette section par une analyse comparative des différents modèles de contexte existants.

2.3.1. La notion de contexte : définitions, caractéristiques et dimensions

La notion de « *contexte* » est un concept très large, exploré depuis plusieurs années dans plusieurs domaines de recherche tel que l'Intelligence Artificielle et l'Informatique Pervasive (Kirsch-Pinheiro, 2006). Cette notion permet, entre autres, de mieux comprendre les interactions des utilisateurs mobiles avec le système et leurs attentes vis-à-vis du même système. La notion de contexte est ainsi utilisée par certains systèmes dans leur processus d'adaptation. Ce processus d'adaptation permet de fournir une réponse plus appropriée à l'utilisateur, potentiellement répondant au mieux à ses besoins dans un contexte donné. Dans cette perspective, la notion de *contexte* apparaît comme un élément central dans une démarche d'adaptation dans un tel système (Najar et al., 2009). Au regard de la littérature, nous pouvons constater qu'elle fait par ailleurs l'objet de nombreuses définitions et interprétations. Plusieurs travaux de recherche (Schilit et Theimer, 1994) (Brown et al., 1997) (Dey, 2001) (Strang et Linnhoff-Popien, 2004) se sont focalisés sur la définition et l'utilisation de contexte. Cette section aborde les différents travaux et les définitions portés sur la notion de contexte, ainsi que ses différentes caractéristiques et dimensions proposées dans la littérature.

2.3.1.1. Définitions

Afin d'utiliser efficacement la notion de contexte, il faut commencer par *comprendre la signification de contexte* et *comment il peut être utilisé* (Dey, 2001). La définition précise de contexte et de ses caractéristiques s'est avérée une question assez délicate. En effet, la notion de contexte est utilisée et interprétée différemment selon le domaine de recherche dans lequel elle est employée et selon les perspectives offertes.

Les premiers travaux dans les Systèmes Sensibles au Contexte ont proposé une vision et une utilisation particulièrement limitée du contexte. Initialement, de nombreux chercheurs, tels que (Schilit et Theimer, 1994) (Brown et al., 1997) (Ryan et al., 1997), représentent la définition de contexte comme une énumération des différents types d'informations portées sur l'utilisateur ou l'environnement dans lequel s'intègre l'application et qui sont jugés pertinents.

Schilit et Theimer (Schilit et Theimer, 1994) présentent l'une des premières tentatives de formulation de la notion de contexte. Ces auteurs limitent la définition de contexte à l'observation de *la localisation de l'utilisateur, de l'ensemble des utilisateurs présents, des objets à proximités, et des changements apportées à ces éléments* (Schilit et Theimer, 1994). Schilit et al. (Schilit et al., 1994) ne cherchent pas à comprendre la nature du contexte. Ils affirment tout simplement que les aspects les plus importants de la notion de contexte peuvent être déterminés en répondant aux questions « *où se trouve l'utilisateur ?* », « *avec qui se trouve-t-il ?* » et « *quelles sont les ressources à proximité ?* ». Une autre définition est ensuite proposée par Brown et al. (Brown et al., 1997). Ces derniers considèrent que le *contexte regroupe la localisation de l'utilisateur, les identités des personnes qui l'accompagnent, le temps représentant un moment de la journée, la saison, la température, etc.* (Brown et al., 1997). Pour Ryan et al. (Ryan et al., 1997), qui appliquent la sensibilité au contexte dans le cas d'un logiciel de prise de note pour l'archéologie, *le contexte se définit de façon plus générale comme la localisation, l'environnement, l'identité et le temps relatifs à l'utilisateur*. Toujours dans le même principe de définir la notion de contexte en énumérant ses différents types d'information, Dey (Dey et al., 1998) apportent une autre définition qui présente le contexte comme étant *l'état émotionnel de l'utilisateur, le focus de l'attention, la localisation et l'orientation, la date et le temps, les objets et les personnes dans l'environnement de l'utilisateur* comme les éléments constituant la définition de contexte. Enfin, Chen et Kotz (Chen et Kotz, 2000) définit le contexte comme *l'ensemble des états et des paramètres environnementaux qui soit déterminent le comportement d'une application soit dans lesquels un événement d'application se produit et est intéressant pour l'utilisateur.*

Nous observons que ces premiers travaux de recherche se sont focalisés plus précisément sur l'identification des éléments décrivant le contexte d'usage d'un système, plutôt que de comprendre sa réelle signification. Ils demeurent assez vagues sur la nature de la notion de contexte elle même. Toutefois, suite à ces premiers travaux, une plus grande attention a été accordée à la notion de contexte. Pascoe 1998 (Pascoe, 1998) et Dey (Dey, 2001), par exemple, ont proposé des définitions de contexte plus générales et largement acceptées. Ces définitions donnent un sens plus opérationnel à la notion de contexte, employée pour le

développement d'applications pervasives (ubiquitaires) et mobiles. Leur objectif est d'abstraire les précédents travaux sur le contexte de leurs liens aux expérimentations et aux scénarii bien spécifiques. Ainsi, Pascoe 1998 (Pascoe, 1998) voit le *contexte comme le sous-ensemble mesurable d'un environnement porté à l'intérêt des utilisateurs*. Cette définition ne se limite pas à un domaine en particulier et a comme objectif de pouvoir être dérivée et applicable à différents types d'application et de scénarii. Par contre, de part sa généricité, cette définition est parfois difficilement applicable à un cas concret. Quelques années plus tard, apparaît la définition la plus connue proposée Dey (Dey, 2000) qui décrit le contexte comme étant « *toute information qui peut être utilisée pour caractériser la situation d'une entité. Une entité est une personne, un endroit ou un objet considérés comme pertinents pour l'interaction entre un utilisateur et une application, y compris l'utilisateur et les applications elles-mêmes* ». Cette définition a été illustrée et utilisée par la suite comme une référence dans plusieurs travaux de recherche. La pertinence de cette définition émerge de sa capacité à être dérivée selon différents degrés de granularité, en plus d'être utilisée dans n'importe quel scénario. Elle concerne plus particulièrement la conception des Systèmes Sensibles au Contexte, puisqu'elle prend en compte la pertinence des éléments pour les interactions entre l'utilisateur et le système. Cet auteur (Dey, 2001) estime d'ailleurs que les précédentes définitions étaient trop précises, car il est impossible d'énumérer les aspects qui sont importants pour toutes les situations, ceux-ci pouvant changer d'une situation à une autre.

La notion de contexte s'est étendue avec (Mostefaoui et al., 2004) (Brézillon, 2005) (Kirsch-Pinheiro, 2006). Désormais, elle ne se limite plus à l'utilisateur qui a effectué une action ou à sa localisation, mais elle s'étend aux informations physiques (localisation, temps, etc.), aux informations sociales et même organisationnelles (rôle de l'utilisateur, etc.).

D'une manière plus générale, Mostefaoui et al. (Mostefaoui et al., 2004) définissent le contexte comme *ce qui entoure le centre d'intérêt de l'individu et qui apporte des informations additionnelles capables d'aider à la compréhension de ce centre d'intérêt*. Dans ce même cadre, Brézillon (Brézillon, 2005) précise que le contexte dépend également de l'activité courante de l'utilisateur. L'activité est considérée, par ces auteurs, comme un élément clé de contexte : elle détermine les informations, les connaissances, les objets de l'environnement, etc. qui représentent les éléments les plus pertinents autour d'elle et qui sont nécessaires à son accomplissement. Selon cet auteur, le contexte entoure un *focus* (par exemple, la tâche à accomplir ou l'interaction) et donne un sens aux éléments liés à ce focus sans intervenir explicitement dans celui-ci. La notion de contexte est ainsi employée afin d'orienter le focus d'attention, à savoir les sous-ensembles d'un terrain d'entente jugé pertinent pour l'activité en cours (Brézillon, 2005). Sous cette optique, nous pouvons voir le contexte comme étant l'ensemble des caractéristiques de l'environnement dans lequel se déroule l'activité, mais qui sont séparés de l'activité elle-même.

Toutefois, malgré le fait que la définition proposée par Dey (Dey, 2000) soit une référence dans le domaine de l'informatique sensible au contexte, elle demeure débattue (Tamminen et al., 2004). Certains auteurs, comme (Greenberg, 2001), la trouvent trop générale et révèlent un problème important d'adaptation au processus de conception. D'autres auteurs, dont

(Chaari et al., 2005), trouvent que le problème essentiel dans cette définition est l'identification des éléments composant la notion de contexte. De plus, selon ces auteurs, la définition proposée par Dey ne distingue pas les données contextuelles des données de l'application. Ces auteurs considèrent que cette séparation est très importante pour la modélisation de contexte. Selon Chaari et al. (Chaari et al., 2004), la notion de contexte correspond à *un ensemble de paramètres qui sont externes à l'application et qui influencent le comportement de celle-ci en définissant de nouvelles vues sur ses données et ses services.* Dans la même ligne, Gensel et al. (Gensel et al., 2008) définissent la notion de contexte comme étant *l'ensemble des caractéristiques de l'environnement physique ou virtuel qui affecte le comportement d'une application et dont la représentation et l'acquisition sont essentielles à l'adaptation des informations et des services*. Elle devient un élément clé de l'Informatique Pervasive, car elle est au centre des mécanismes d'adaptation prônés par les Systèmes Sensibles au Contexte.

Selon cet aperçu de la littérature, nous pouvons observer que le contexte a été défini de multiples façons, selon des points de vue différents. Ceci a engendré les multiples définitions de contexte présentées ci-dessus. Toutefois, la définition proposée par (Dey, 2000) demeure la référence dans le domaine de l'informatique sensible au contexte.

2.3.1.2. Caractéristiques

La notion de contexte expose un certain nombre de caractéristiques dans l'Informatique Pervasive. A partir des travaux de (Henricksen et al., 2002) (Gu et al., 2004) (Baldauf et al., 2007), nous soulignons ici celles qui nous semblent les plus pertinentes :

- **Hétérogénéité** : Les informations de contexte peuvent provenir de diverses sources (Gu et al., 2004). Elles peuvent être capturées (à travers des capteurs physiques, logiques ou virtuels), dérivées (en se basant sur des mécanismes de raisonnement ou de transformation), statiques (qui ne varient pas avec le temps), ou fournies par l'utilisateur (décrivant son profil). De plus, le contexte se définit par toute sorte d'information capable de décrire les objets physiques, les applications et les utilisateurs dans différents domaines. Ceci conduit à une *hétérogénéité* remarquable des informations contextuelles ;
- **Statique vs dynamique** : Les informations de contexte peuvent être définies en tant qu'informations statiques ou dynamiques de contexte. Cette caractéristique dépend du niveau de variabilité des valeurs de cette information contextuelle. D'une part, les informations dites *statiques* sont celles qui décrivent les aspects ne variant pas avec le temps, tel que la date de naissance de l'utilisateur. D'autre part, les informations dites *dynamiques* sont celles qui varient avec le temps selon les changements capturés, tel que la localisation d'un utilisateur. En effet, les Systèmes Sensibles au Contexte se caractérisent par leurs changements fréquents, ce qui implique que la majorité des informations contextuelles est dynamique. Brézillon (Brézillon, 2002), Rey et Courtaz (Rey et Coutaz, 2004) ont souligné ce caractère dynamique, dit aussi évolutif, de la notion de contexte. Selon Brézillon et al. (Brézillon, 2002), le contexte doit être

considéré comme étant un espace qui dépend fortement de la situation et qui est en continuelle évolution. Pour Chaari et al. (Chaari et al., 2004), les paramètres qui composent le contexte évoluent durant l'exécution du système ;

- **Interdépendance** : Les informations de contexte peuvent être dépendantes entre elles (Henricksen et al., 2002)(Gu et al., 2004). Certaines informations peuvent être interdépendantes par raisonnement et par dérivation de règles par exemple. En effet, par raisonnement sur le contexte, une information peut être inférée à partir d'une ou de plusieurs autres informations capturées, définies ou agrégées. Cette dérivation démontre certaines dépendances internes entre les éléments de contexte. Par exemple, dans certains cas, l'activité de l'utilisateur ne peut être dérivée que de sa localisation et de son planning numérique ;
- **Imperfection** : L'acquisition des informations contextuelles se fait à travers divers capteurs qui peuvent fournir des informations défectueuses ou être même la déconnection ou l'échec de communication entre le fournisseur et le consommateur de contexte, peuvent conduire à l'imperfection des éléments de contexte (Henricksen et al., 2002). En effet, certaines informations contextuelles peuvent être *incorrectes* si elles ne reflètent pas la situation réelle de ce qu'elles modélisent. De plus, elles peuvent être *incomplètes* dans le cas où certains éléments de contexte demeurent inconnus. Ceci est dû, par exemple, à l'arrêt d'un capteur ou à un échec de communication entre le fournisseur et le consommateur de contexte. En outre, certaines informations de contexte peuvent être *inconsistantes* si elles contiennent des informations contradictoires par exemple.

La prise en compte de la notion de contexte pour gérer la caractéristique dynamique d'un Système Sensible au Contexte dépend de la nature de chaque type d'éléments de contexte, de sa pertinence et de son utilisation dans le système. Ainsi, étant donnée la diversité des informations de contexte, il est utile de bien les classifier par dimension afin de faciliter leur utilisation et compréhension. C'est pour cette raison que nous présentons dans la section suivante les dimensions potentielles de contexte.

2.3.1.3. Dimensions

Les difficultés rencontrées à définir d'une manière unique et non ambiguë la notion de contexte ont encouragé différentes interprétations des dimensions. Ces dimensions regroupent les informations contextuelles qui appartiennent à une même catégorie. En d'autres termes, ils représentent la classification des éléments de contexte afin de faciliter leur interprétation, leur compréhension ainsi que leur utilisation. Il est possible de classifier les éléments de contexte de différentes manières. Henricksen et al. (Henricksen et al., 2002) notent que la catégorisation des éléments de contexte en un ensemble de dimensions est utile pour gérer la qualité de contexte. En outre, cette classification est également utile pour la modélisation (Soylu et al., 2009), que nous discuterons dans la section 2.3.2.

La description de ces dimensions a fait l'objet de plusieurs propositions résumées dans multiples travaux (Baldauf et al., 2007) (Banâtre et al., 2007) (Han et al., 2008) (Soylu et al.,

2009). Ces travaux proposent des dimensions de contexte représentées à un niveau élevé de granularité. Cette classification des éléments de contexte, selon Soylu et al. (Soylu et al., 2009), permet une stratification claire pour le développement des Systèmes Sensibles au Contexte. Elle peut servir comme un premier pas vers une conceptualisation générique des éléments de contexte. Nous présentons certaines de ces dimensions :

- **Dimension temporelle** caractérise, selon (Soylu et al., 2009), des éléments de contexte dont l'existence ou l'importance dans la perception du contexte est liée à un moment donné. Cette dimension peut être utilisée pour décrire le temps associé aux différents types d'information de contexte, par exemple : le fuseau horaire, l'heure actuelle de l'utilisateur, le début et la fin d'une situation, la durée d'un évènement, d'une activité, ainsi que d'un planning de travail, ou d'utilisation d'une ressource ... Par ailleurs, le temps est un élément très important pour établir et gérer l'historique des contextes ou des situations passées permettant d'enrichir le contexte ;
- **Dimension spatiale** définit, selon (Soylu et al., 2009), la localisation. Cette localisation peut déterminer un espace précis dans lequel s'organise certains dispositifs ou objets de l'environnement, comme elle peut décrire l'emplacement des utilisateurs qui se déplacent d'un endroit à un autre. Cette dimension regroupe la *localisation physique*, tels que la position absolue (adresse géographiques), le lieu (à la maison, chez le client, etc.), coordonnées GPS, ainsi que la *localisation virtuelle,* tel que l'adresse IP considérée comme une localisation dans un réseau ;
- **Dimension relative aux dispositifs** permet de mesurer les caractéristiques de la plateforme cliente (Van Welie et De Ridder, 2001) (Groot et Welie, 2002). Cette dimension est appropriée dans le cas où le système doit s'adapter aux capacités et aux conceptions très hétérogènes de ces dispositifs. Cette dimension caractérise les *dispositifs mobiles* (*i.e. smartphone*, ordinateur portable, tablette, etc.), les *dispositifs fixes* (*i.e.* écrans LCD, des capteurs fixes, des hauts parleurs, etc.), ainsi que d'autres *ressources informatiques*. Les caractéristiques de ces dispositifs (taille, résolution, puissance de calcul, etc.) peuvent varier beaucoup, ainsi que leur disponibilité. Par conséquent, le système doit adapter son contenu en fonction des caractéristiques de ces dispositifs. Les dispositifs fixes peuvent communiquer et échanger des données avec d'autres types de dispositifs. Pour les ressources informatiques, certains systèmes prennent en compte la charge et la puissance du processeur de l'ordinateur, des périphériques, la charge du réseau en fonction de sa bande passante, etc. ;
- **Dimension relative à l'utilisateur** représente les informations de contexte relatives à un représentant du public cible du système, décrit par ses capacités physiques et cognitives (*i.e.* profil de l'utilisateur, les personnes à proximité, la situation sociale actuelle, etc.) (Schilit et al., 1994) (Chen et Kotz, 2000) ;
- **Dimension relative à l'environnement** regroupe les informations contextuelles portant sur les informations périphériques à la tâche de l'utilisateur, mais susceptibles de l'influencer (Calvary et al., 2002). Pour certains systèmes, il est intéressant voire nécessaire de mesurer les caractéristiques de l'environnement autour de l'utilisateur afin de s'adapter et de réagir en conséquence. Par exemple, en mesurant le niveau de bruit, nous pouvons ajuster le niveau sonore du haut parleur d'un dispositif mobile ;

- **Dimension relative à l'infrastructure** caractérise les impacts de la communication entre les composants distribués sur l'activité fonctionnelle du système et sur l'interaction avec le ou les utilisateur(s) (Rodden et al., 1998). Selon Rodden et al. (Rodden et al., 1998), dans les systèmes mobiles la nature de l'infrastructure est susceptible de changer pendant son usage. Cette variabilité dans l'infrastructure peut avoir un impact sur l'interaction. Il est essentiel que les styles et les interfaces des interactions soient compatibles avec l'état de l'infrastructure. Les propriétés particulières de l'infrastructure, comme la topologie du système, doivent ainsi être intégrées lors de la conception d'un système mobile ;
- **Dimension relative au système** évalue l'utilisation des ressources, par exemple la mémoire, le processeur et le réseau, des composants du système et des capacités du dispositif mobile de l'utilisateur.

Malgré ces différentes définitions et visions de la notion de contexte, l'informatique sensible au contexte a su adopter des démarches pragmatiques pour proposer différentes modélisations de contexte. En effet, la gestion de contexte et de ses différentes caractéristiques nécessite de représenter le contexte explicitement dans le système. Nous décrivons dans la section suivante la modélisation de cette notion de contexte.

2.3.2. Modélisation de contexte

Dans les Systèmes Sensibles au Contexte, la notion de contexte joue un rôle central qui guide le mécanisme d'adaptation utilisé pour personnaliser le contenu et les services en conséquence. La façon dont les informations de contexte sont utilisées dans ces systèmes dépend de : (i) quelle information est observée ; et (ii) comment elle est représentée. Pour (Brézillon, 2002), *une représentation efficace du contexte en machine, tant en termes de modélisation de connaissances que de raisonnement à partir de celles-ci, est un problème à résoudre, et ce, aussi bien du point de vue de la programmation que de son utilisation*. Un modèle de contexte est une représentation explicite de l'information de contexte dans le système afin qu'il puisse stocker, interpréter, gérer et raisonner sur ces informations contextuelles pour un objectif précis. Dans un environnement pervasif, la modélisation de ces informations contextuelles dans un système est nécessaire car elle permet de gérer la sensibilité au contexte et l'adaptation. En d'autres termes, les capacités d'adaptation d'un Système Sensible au Contexte dépendent du modèle de contexte utilisé (Najar et al., 2009). Ainsi, un modèle de contexte bien conçu est le fondement d'un Système Sensible au Contexte (Strang et Linnhoff-Popien, 2004). De toute évidence, le formalisme choisi pour représenter ce modèle est important, car il détermine les méthodes de raisonnement que le système peut utiliser pour effectuer certaines adaptations. Grâce à la littérature, nous pouvons observer que de nombreux modèles de contexte ont été proposés par la communauté de recherche (Strang et Linnhoff-Popien, 2004) (Najar et al., 2009) (Bettini et al., 2010). Chaque modèle présente différents points de vue de la notion de contexte qui ont été étudiés dans différents domaines d'application (*e.g.* intelligence ambiante, systèmes mobiles de tourisme). Un modèle de contexte assure la définition de processus d'adaptation indépendant et isole ce processus des

techniques d'acquisition de contexte, représentant ainsi la première exigence pour la maintenance et l'évolution des Systèmes Sensibles au Contexte (Najar et al., 2009).

L'évolution des Systèmes Sensibles au Contexte de la dernière décennie a été suivie par une importante évolution des modèles de contexte, allant des simples structures clé-valeur aux modèles basés sur des ontologies. Les approches existantes de modélisation de contexte diffèrent ainsi par la puissance d'expression des modèles de contexte, par le support qu'ils peuvent fournir pour raisonner sur des informations de contexte, et par la performance de calcul de ce raisonnement. Strang et Linnhoff-Popien (Strang et Linnhoff-Popien, 2004) et Bettoni *et al.* (Bettini et al., 2010) ont souligné les approches les plus pertinentes. Ces auteurs classifient les modèles de contexte en fonction de leurs structures de données utilisées pour maintenir et échanger les informations contextuelles dans un système donné. Dans les prochaines sections, nous présentons les structures de données les plus couramment utilisées.

2.3.2.1. Modélisation de contexte basée sur les paires clé-valeur

La modélisation de contexte constituée de paires *« clé-valeur »* correspond au formalisme le plus simple pour représenter le contexte. Il s'agit de représenter le contexte d'utilisation comme un ensemble de *paires* contenant chacune une *clé* et la *valeur* qui lui correspond. Dans cette approche, un élément observé de l'environnement est considéré comme une *clé*, et la *valeur* de cet élément représente les données de l'information contextuelle. Schilit et Theimer (Schilit et Theimer, 1994) se basent sur ce formalisme pour modéliser des informations de contexte tel que la localisation. Cette représentation est également adoptée par le *Context Toolkit* (Salber et al., 1999) (Dey, 2000).

Les approches de modélisation constituée des paires *clé-valeur* se caractérisent par leur simple représentation, leur gestion facile des paires (dans le cas d'un nombre raisonnable de paire) et leur facilité de stockage. La simplicité des pairs clé-valeur peut être un avantage d'un point de vue gestion, mais ils représentent un inconvénient majeur lors de l'interprétation sémantique et si le critère d'ambiguïté est à considérer. La modélisation par les paires *clé-valeur* ne permet de représenter que la valeur capturée pour un élément observé de l'environnement. Avec cette simple modélisation, la qualité des informations contextuelles n'est pas prise en compte. Ce type de modélisation ne permet pas de déterminer si les informations contextuelles capturées sont incomplètes, ambiguës ou même incertains. De ce fait, l'ambiguïté des informations contextuelles ne peut pas être prise en compte par ce modèle. De plus, cette modélisation manque de capacités pour des structurations assez sophistiquées. En effet, plus le nombre de paires possibles évolue, plus difficile sera la gestion de ces paires et des aspects sémantiques liés à ces paires. Cette augmentation entraine avec elle un risque d'avoir des doublons, entrainant ainsi une incohérence entre les paires contenant des informations sur un même élément de contexte. Cette modélisation est également confrontée à un autre problème, à savoir la difficulté de sa réutilisation dans d'autres systèmes que celui d'origine, car l'interprétation des paires clé-valeur dépend de l'application. Ceci entraine un problème d'interopérabilité si les clés et les valeurs ne sont pas standardisées.

2.3.2.2. Modélisation de contexte basée sur les schémas de balisage

Une seconde approche soulignée dans la littérature est l'approche utilisant les modèles de schéma de balisage. Ces modèles formalisent les informations de contexte selon une structure de données fixe et hiérarchique (un arbre syntaxique). Cette structure est constituée de balises avec des attributs et de contenu permettant d'exprimer des relations plus complexes tels que les associations. Cette approche de modélisation de contexte est sérialisée dans des documents XML ou RDF (Lassila et Swick, 1999) afin d'améliorer l'interopérabilité.

Ces modèles de contexte sont souvent utilisés pour représenter des informations statiques portant sur des « *profils* » d'entités. Ils peuvent formaliser par exemple des *profils de l'utilisateur*, tel que l'approche *Friend-Of-A-Friend* (FOAF) (Brickley et Miller, 2005), et des *profils des dispositif*, tels que les approche *Composite Capabilities/Preferences Profile* (CC/PP) (Klyne et al., 2004) (Lemlouma, 2004) (Kiss, 2010), *User Agent Profile* (UAProf) (WAP FORUM, 2006) et *Comprehensive Structured Context Profile* (CSCP) (Buchholz et al., 2004), qui ont pu atteindre un certain niveau d'expressivité par la sérialisation XML et RDF.

Par exemple, le standard CC/PP (Klyne et al., 2004) est une recommandation W3C permettant de créer des « *profils* » décrivant non seulement les caractéristiques logicielles et matérielles d'un dispositif, mais également les préférences de l'utilisateur. Un profil CC/PP est utilisé pour personnaliser le contenu et adapter la présentation sur la base des capacités des dispositifs et des préférences des utilisateurs. Chaque profil est constitué d'un ensemble d'attributs et de valeurs associées. Le standard CC/PP est lui-même basé sur un autre standard, RDF (Manola et al., 2004). Le point important que relève l'utilisation de RDF, par rapport à l'utilisation d'un ensemble de paires clé/valeur, est la possibilité de décrire des métadonnées sur ces paires et de définir un vocabulaire commun réunissant les propriétés susceptibles d'être décrites.

Plus spécifiquement, Lemlouma (Lemlouma, 2004) propose un modèle de contexte se basant sur CC/PP pour représenter le contexte d'utilisation. Ce modèle est utilisé par son architecture NAC (*Negociation Adaptation Core*) qui assure, dans un environnement hétérogène, la transmission au client d'un contenu dont la présentation est adaptée aux contraintes des dispositifs mobiles. Cet auteur utilise le modèle CC/PP pour décrire les capacités physiques (capacité mémoire, taille écran, etc.) et logicielles (systèmes d'exploitation, navigateur, etc.) d'un terminal. La Figure 2 illustre un exemple de modélisation de profil CC/PP, présenté par Lemlouma et al. (Lemlouma, 2004). Cet exemple décrit les capacités d'affichage d'un terminal en particulier, à travers les valeurs de l'élément *HardwarePlatform* (largeur 320 et hauteur 200), ainsi que la description du système d'exploitation et du navigateur Web installé dans le dispositif (élément *SoftwarePlatform*). Dans ce travail, Lemlouma et al. 2004 (Lemlouma, 2004) propose deux mécanismes de capture (extraction de contexte) à savoir l'extraction statique qui fait appel à une interrogation paramétrée d'une base des profils ou l'extraction dynamique qui calcule la valeur de certaines caractéristiques de l'environnement. Cet auteur propose dans ce cadre un mode d'interaction optimisé avec un répertoire de profil permettant de maintenir une base des profils CC/PP.

```
<ccpp:component>
  <rdf:Description
      rdf:about="http://www.example.com/profile#TerminalHardware">
    <rdf:type
        rdf:resource="http://www.example.com/schema#HardwarePlatform" />
    <ex:displayWidth>320</ex:displayWidth>
    <ex:displayHeight>200</ex:displayHeight>
  </rdf:Description>
</ccpp:component>

<ccpp:component>
  <rdf:Description
      rdf:about="http://www.example.com/profile#TerminalSoftware">
    <rdf:type
        rdf:resource="http://www.example.com/schema#SoftwarePlatform" />
    <ex:name>EPOC</ex:name>
    <ex:version>2.0</ex:version>
    <ex:vendor>Symbian</ex:vendor>
  </rdf:Description>
</ccpp:component>
```

Figure 2. Exemple de profil CC/PP (d'après (Lemlouma, 2004))

Après l'apparition de CC/PP, plusieurs autres extensions de ce standard ont été proposées. Le forum WAP, par exemple, (WAP FORUM, 2006) a proposé UAProfile come une autre approche de modélisation qui adopte la spécification de CC/PP. A l'instar de CC/PP, un profil UAProf présente une hiérarchie à deux niveaux composée d'éléments et de leurs propriétés. En revanche, contrairement à CC/PP, la spécification UAProf propose également un vocabulaire concret, représenté par un ensemble spécifique d'éléments et d'attributs, pour décrire la prochaine génération de téléphones WAP. Une autre extension de ce langage CC/PP a été proposée par Held et al. (Held et al., 2002). Cette extension nommée CSCP (*Comprehensive Structured Context Profiles*) ne définit aucune hiérarchie fixe, au contraire de la modélisation de CC/PP. Le CSCP s'appuie sur la flexibilité de RDF/S pour exprimer la structure naturelle des informations d'un profil requises pour l'information contextuelle. Cette modélisation de contexte est un métalangage basé aussi sur RDF et hérite de ce dernier, l'interopérabilité, la décomposition et l'extensibilité. Il est plus flexible que CC/PP concernant la structuration des documents et étend le mécanisme pour exprimer les préférences de l'utilisateur.

Par rapport aux modèles de paires clé-valeur, les modèles de schéma de balisage fournissent un langage plus expressif pour structurer l'information de contexte qui peut, dans certain cas, être adapté à un domaine d'application plus spécifique.

2.3.2.3. Modélisation de contexte graphique et orientée objet

L'évolution de la modélisation de contexte se poursuit avec l'émergence de *modèles graphiques* (ORM, CML, UML, etc.) et des modèles orientés objets, dont la force est leur structure. Une des propositions les plus pertinentes dans cette approche est le langage de modélisation de contexte (CML) (*Context Modelling Language),* décrit au départ par Henricksen et al. (Henricksen et al., 2002) et affiné dans des travaux ultérieurs (Henricksen et Indulska, 2004)(Henricksen et Indulska, 2006). CML est une approche de modélisation

graphique de contexte destinée à la modélisation des bases de données. CML représente une extension de la modélisation ORM (*Object-Role Modeling*) (Halpin, 2001) qui est une approche basée sur les « *faits* ». Cette approche (ORM), qui a été conçue pour la modélisation conceptuelle des bases de données, modélise les faits représentant les informations vraies ou correctes dans une application, et les types des faits définissant les types de l'information.

CML fournit une notation graphique (visible sur la Figure 3) conçue pour soutenir le concepteur dans l'analyse et la spécification formelle des exigences d'une application sensible au contexte. Ce langage propose des constructions de modélisation pour : la capture des différentes classes et sources de faits, la capture des informations imparfaites en utilisant des métadonnées de qualité, la capture des dépendances entre les types des faits, la capture des historiques de certains types de faits et des contraintes sur ces historiques (Bettini et al., 2010). Les concepts modélisés fournissent une base formelle pour la représentation et le raisonnement sur certaines propriétés de l'information contextuelle (persistance, qualité, interdépendances, etc.). Chaque entité modélisée décrit un objet physique ou conceptuel, comme une personne, un dispositif ou un moyen de communication. Les attributs représentent les propriétés des entités auxquelles ils sont attachés par le biais des associations. Les associations connectent également les entités entre elles.

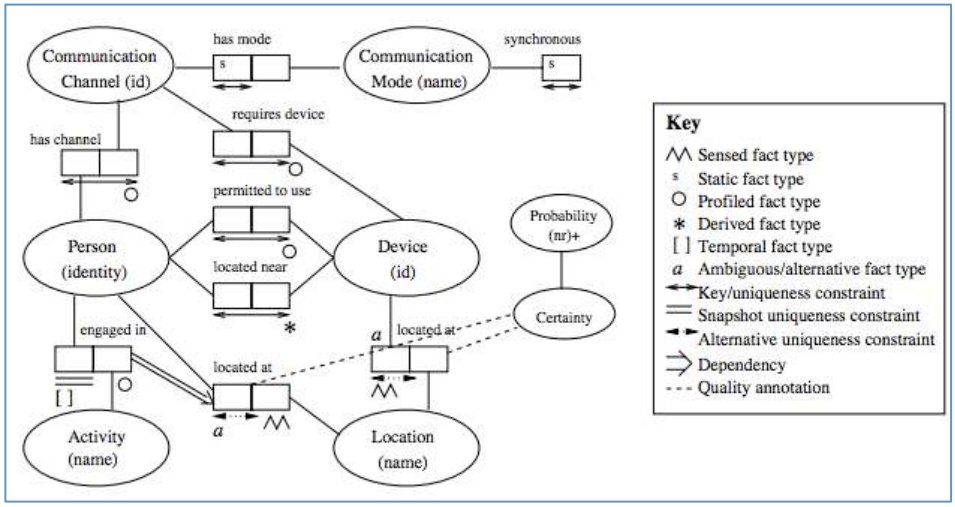


Figure 3. Exemple de modélisation CML (d'après (Henricksen et al., 2002))

La Figure 3 illustre un exemple d'une modélisation de contexte selon le modèle CML (Henricksen et Indulska, 2006). Le modèle représente les utilisateurs (*Person*), leurs activités (*Activity*), les terminaux utilisés (*Device*), la localisation des utilisateurs et des terminaux (*Location*), les canaux de communication (*Communication Channel*) et les modes de communication (*Communication Mode*). Chaque ellipse de la Figure 3 représente un type d'objet avec la valeur entre parenthèses décrivant le schéma de représentation utilisé pour le type d'objet, tandis que chaque case représente un rôle joué par un type d'objet dans un type de fait. Ainsi les activités sont associées à un fait temporel. La localisation d'un utilisateur ou d'un dispositif est une information capturée et a une valeur de certitude associée correspondant à un indice de confiance en fonction de l'endroit de capture.

La modélisation CML permet d'interroger des informations incertaines (ambiguës) en utilisant une logique à trois valeurs, en plus des assertions et des conditions complexes exprimées en utilisant une forme de logique de prédicat (Henricksen et Indulska, 2006). Le point faible de cette approche est sa modélisation des informations jugées « *plates* » par Bettini et al. (Bettini et al., 2010). De plus, même si CML peut être utilisé pour le développement d'une application en particulier, il ne fournit pas de support pour l'interopérabilité entre applications.

Par ailleurs, nous observons dans la littérature l'émergence des modélisations de contexte basées sur les *modèles orientés objets*. Ces modèles peuvent être visualisés en utilisant UML. Ces diagrammes reposent sur une méthode de notation standard et générique. Grâce à sa structure générique, UML est approprié pour modéliser le contexte selon un ensemble de classes, d'objets et d'associations. Ceci est illustré, par exemple, dans Bauer (Bauer, 2003), dans lequel les aspects contextuels pertinents à la gestion du trafic aérien sont modélisés comme des extensions UML.

Les approches de modélisation de contexte orientée objets tirent profit de l'encapsulation et de la réutilisation propre l'approche objets afin de couvrir une partie des problèmes liés à la dynamique de la gestion de contexte dans les environnements pervasifs. Les détails concernant le traitement des éléments de contexte sont encapsulés au niveau de l'objet, et ainsi ils sont masqués aux autres composants. Selon (Bouzy et Cazenave, 1997), la modélisation objets permet de *définir le plus petit nombre de propriétés, fonctions et règles [...] afin de simplifier la représentation des connaissances dans des domaines et de systèmes très complexes.* Par exemple, Kirsch-Pinheiro et al. (Kirsch-Pinheiro et al., 2004) (Kirsch-Pinheiro, 2006) proposent une approche orientée objets pour la structuration des éléments de contexte et de leurs relations. Un tel modèle est utilisé pour personnaliser le contenu fourni par des systèmes collaboratifs basés sur le Web : le contenu fourni est sélectionné selon le contexte et les préférences de l'utilisateur. L'originalité de ce modèle est la proposition d'éléments de contexte qui sont liés aux aspects collaboratifs (rôle de l'utilisateur, activités, etc.) en plus des aspects physiques (la localisation de l'utilisateur, le dispositif, etc.). Néanmoins, la capture et la maintenance de ces éléments de contexte sont en dehors du modèle. Les auteurs (Kirsch-Pinheiro et al., 2004) supposent l'existence de composants externes qui observent les éléments de contexte correspondants et alimentent ainsi le modèle.

Toutefois, les modèles de contexte orientées objets demeurent peu adaptées au partage des connaissances dans un environnement ouvert et dynamique. Elles nécessitent certains accords d'exécution de bas niveau entre les applications pour assurer l'interopérabilité.

2.3.2.4. Modélisation de contexte basée sur les ontologies

Dans le cadre de la modélisation de contexte, nous observons l'émergence de nouvelles approches impliquant la modélisation sémantique, en fournissant une description plus structurée et plus riche de contexte basée sur les *ontologies*. Les ontologies, selon Grubber (Gruber, 1993) et Uschold (Uschold et al., 1996), se basent sur l'idée de spécifier un

vocabulaire décrivant un ensemble de concepts et les relations pertinentes entre eux. L'ontologie représente ainsi une spécification explicite d'une conceptualisation. Plus spécifiquement, dans la modélisation de contexte, les ontologies fournissent une description formelle et sémantique des informations de contexte en termes d'objets, concepts, propriétés et relations. Elles sont largement acceptées pour la modélisation des informations de contexte dans le domaine de l'Informatique Pervasive. La principale raison de leur acceptation est la popularité et la maturité des langages venus du Web sémantique. Dans le Web sémantique, plusieurs langages de spécification et de description des ontologies existent, tels que RDFS/OWL (Beckett, 2004) et OWL (Horrocks et al., 2003). Le standard OWL définit une ontologie comme *une collection d'informations, notamment des informations sur des classes et des propriétés* (Smith et al., 2004). La modélisation à base d'ontologies exploite la puissance de la représentation et du raisonnement des logiques de description pour de multiples raisons énumérées par Bettini et al. (Bettini et al., 2010) : (i) l'expressivité du langage est utilisée pour décrire des données de contexte plus complexes qui ne peuvent pas être représentées en CC/PP ; (ii) en fournissant une sémantique formelle aux données de contexte, il devient ainsi possible de partager et/ou intégrer le contexte de différentes sources ; et (iii) les moteurs de raisonnement disponibles peuvent être utilisés afin de vérifier, d'une part, la consistance de l'ensemble des relations entre les éléments de contexte et de dériver des informations de contexte de plus haut niveau. Ainsi, à part la richesse et, d'autre part, l'expressivité des représentations sémantiques apportées par les ontologies, ces dernières se caractérisent également par leur capacité de partage de connaissance et de réutilisation. Ceci est particulièrement important dans un environnement pervasif caractérisé par son hétérogénéité et son dynamisme.

De ce fait, et dans la perspective d'une modélisation de contexte plus riche et avec un fort contenu sémantique, plusieurs travaux ont choisi l'utilisation d'ontologies (Strang et Linnhoff-Popien, 2004) (Chen et al., 2003) (Gu et al., 2004). Nous détaillons certains de ces travaux dans les sections qui suivent.

2.3.2.4.1. L'ontologie CONON

L'ontologie de contexte CONON (*CONtext ONtology*) est une proposition de Wang et al. (Wang et al., 2004) laquelle repose sur les capacités de partage des connaissances et de réutilisation des ontologies afin de définir ses éléments de contexte. Wang et al. (Wang et al., 2004) se sont concentrés sur la classification et la représentation des éléments de contexte et sur le raisonnement sur ses éléments. Ils ont représenté l'ontologie CONON en OWL-DL sous une forme hiérarchique à deux niveaux. Le plus haut niveau *(upper onotology)* décrit l'ensemble de concepts les plus généraux et qui sont communs à tous les domaines, tels que la localisation, l'activité la personne, etc. Chacune de ces classes est associée à des propriétés afin d'exprimer ses relations avec les autres classes. Le bas niveau *(domain-specific ontologies)* représente une collection d'ontologies, qui définit les détails des concepts généraux et leurs propriétés dans chaque sous-domaine. Ces ontologies représentent des concepts de contexte plus spécifiques qui sont dépendants du domaine. Ce niveau étend les classes abstraites en classes plus spécifiques, permettant ainsi une certaine flexibilité par

l'extension de l'ontologie de haut niveau aux différents domaines.

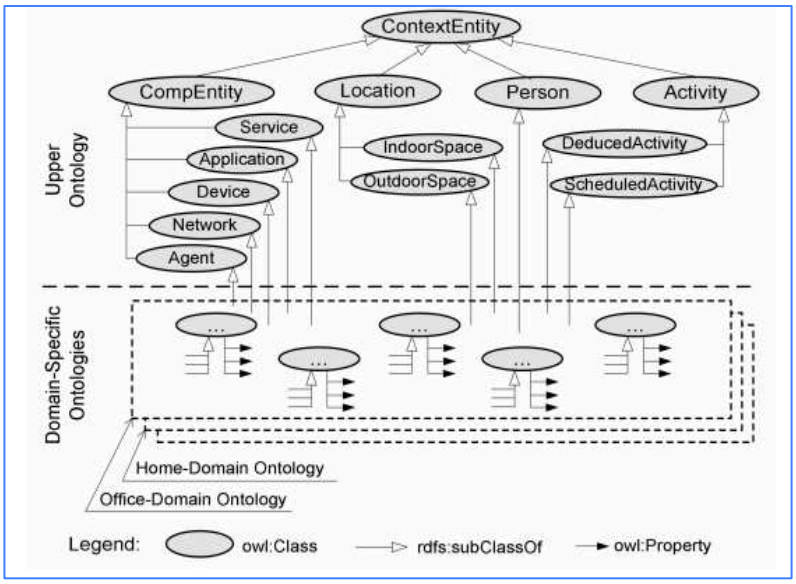


Figure 4. L'ontologie de contexte CONON (d'après (Wang et al., 2004))

Comme l'illustre la Figure 4, Wang et al. (Wang et al., 2004) définissent 4 concepts de base : Personne, Localisation, Activité et Entité de calcul (service, application, réseau, dispositif, etc.). Ainsi, à partir de cette ontologie de haut niveau, il est possible de développer un ensemble d'ontologies qui sont spécifiques à un domaine précis. Pour ce faire, les ontologies dépendantes du domaine peuvent étendre les classes abstraites, définies dans l'ontologie de haut niveau, avec le constructeur *subClassOf* du langage OWL-DL qui va permettre de définir une certaine hiérarchie des classes.

Dans le cadre de ce travail, ces auteurs (Wang et al., 2004) se basent sur un ensemble de règles afin d'exprimer des situations qui sont implémentées avec des prédicats en logique de premier ordre. Ainsi, à travers la création de règles de raisonnement, des connaissances de contexte peuvent être inférées à partir des informations contextuelles de bas niveau. Concrètement, les travaux de Wang et al. (Wang et al., 2004) reposent sur Jena 2 pour le raisonnement sur les ontologies. Alors que les performances d'exécution des méthodes de raisonnement sur le contexte dépendent de la taille de l'ontologie et de la complexité des règles de raisonnement, Wang *et al.* (Wang et al., 2004) concluent qu'un raisonnement puissant basé sur la logique demeure un calcul assez intensif.

2.3.2.4.2. L'ontologie CoDaMoS

L'approche de modélisation de contexte CoDaMoS (*Context-Driven Adaptation of Mobile Services*) (Preuveneers et al., 2004) a été proposée afin de représenter les informations contextuelles utilisées dans un processus d'adaptation et de personnalisation de services selon les capacités des dispositifs et les préférences de l'utilisateur. CoDaMoS représente une ontologie de contexte adaptable et extensible pour la création des infrastructures sensibles au contexte. Comme l'illustre la Figure 5, cette ontologie de contexte est conçue autour de quatre

principaux concepts : *utilisateur*, *service*, *plateforme* et *environnement* qui représentent, selon Preuveneers (Preuveneers, 2009), les aspects les plus importants des informations de contexte. A part ces concepts de base, CoDaMos inclut également des propriétés telles que la localisation courante, les préférences de l'utilisateur, les dispositifs disponibles, etc.

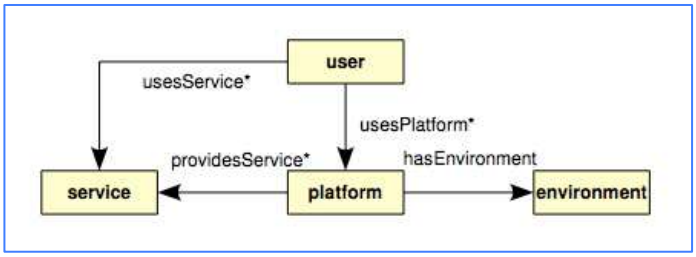


Figure 5. L'ontologie de contexte CoDaMoS (d'après (Preuveneers et al., 2004))

Ces auteurs (Preuveneers et al., 2004) proposent des méthodes de transformation et de raisonnement afin de déduire des informations de contexte de plus haut niveau. La technique utilisée dérive de nouvelles informations contextuelles à partir des règles de dérivation et des faits existants. Ils utilisent Jena 2 pour la description des ontologies et le raisonnement.

2.3.2.4.3. L'ontologie MUSIC

Reichle et al. (Reichle et al., 2008) et Paspallis (Paspallis, 2009) proposent une ontologie de contexte extensible, bien structurée et facile à comprendre. Celle-ci, à l'instar de SOUPA (Chen et al., 2004a) et CONON (Wang et al., 2004), est composée de deux niveaux hiérarchiques. D'une part, le haut niveau définit les éléments de contexte communs à tous les domaines. D'autre part, le bas niveau décrit les éléments de contexte spécifiques à un domaine en particulier.

La Figure 6 illustre la modélisation des éléments de contexte proposée par Reichle et al. (Reichle et al., 2008). L'originalité de cette modélisation repose sur trois concepts de base : (1) l'*entité* à laquelle l'information de contexte se réfère ; (2) le *scope* représentant le périmètre sémantique de l'entité sous forme d'un attribut ; et (3) la *représentation* utilisée en tant que structure interne de l'information de contexte. Ces concepts sont par la suite décrits lors de la modélisation de l'ontologie dans différents domaines d'application.

Cette ontologie de contexte entre dans le cadre du projet IST-MUSIC (IST-MUSIC, 2010), englobant une architecture extensible qui permet de collecter, de stocker, d'organiser et d'accéder aux informations de contexte. Cette architecture repose sur des plug-ins agissant comme des fournisseurs de contexte. Chaque plug-in se charge de la capture dynamique d'une catégorie d'information contextuelle correspondant au triplet <*entité, scope, représentation*> ou de l'interprétation de ces informations (Paspallis, 2009).

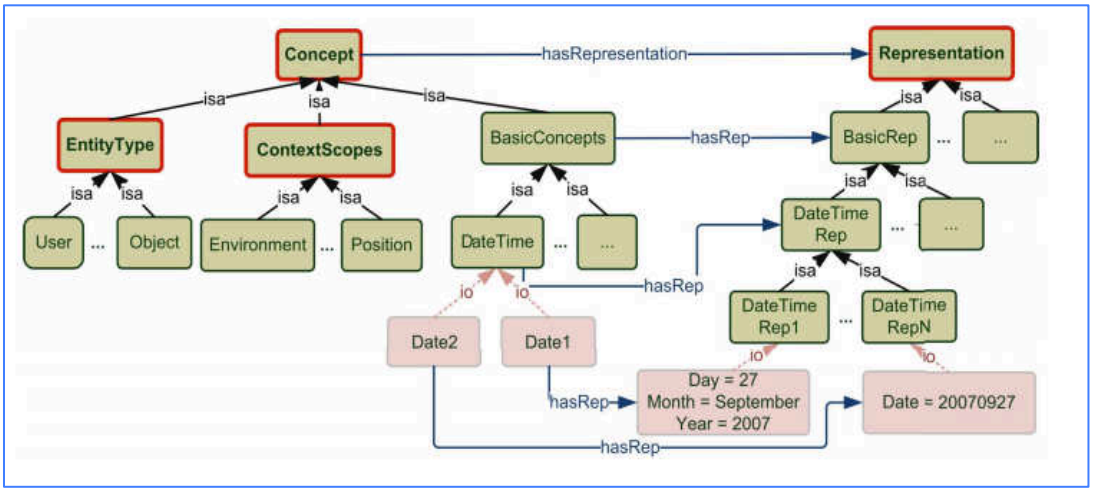


Figure 6. La structure principale de l'ontologie de contexte MUSIC (d'après (Reichle et al., 2008) (Paspallis, 2009))

Finalement, comme nous l'avons mentionné au début de cette section, la prise en compte du contexte dans un Système Sensible au Contexte nécessite l'utilisation d'un modèle de contexte afin de formaliser et de limiter la notion de contexte. La dépendance entre les informations de contexte observées et le comportement d'un Système Sensible au Contexte peut expliquer la grande variété des modèles de contexte (Najar et al., 2009). Par ailleurs, afin de recueillir des informations de contexte, les Systèmes Sensibles au Contexte utilisent généralement des capteurs physiques et/ou logiques. Ces informations sont ensuite interprétées, traitées et mémorisées pour pouvoir réagir à l'environnement en utilisant le modèle de contexte. Ceci représente le processus de gestion de contexte qui sera détaillé dans la section suivante.

2.3.3. Gestion de contexte

Le processus de gestion de contexte est le cœur des Systèmes Sensibles au Contexte. Il est constitué d'un processus itératif qui capture et diffuse les informations contextuelles ainsi que ses changements, les traite et les mémorise. De plus, il permet de prendre des décisions concernant le déclenchement ou non de certaines actions en fonction du contexte observé. Les interactions de l'utilisateur avec le système et avec l'environnement peuvent déclencher des changements du contexte. Il est donc nécessaire que ce processus de gestion de contexte prenne en compte ces changements afin que les processus d'adaptation puissent prendre ces changements en considération.

Selon (Preuveneers, 2009), il y a une nécessité croissante d'applications et de services qui sont plus sensibles aux besoins des utilisateurs, mais moins dépendants de l'attention que cet utilisateur leur porte. La prise en compte d'une large gamme de dispositifs, tels que les téléphones mobiles et les tablettes, représente un facteur essentiel de réussite d'une architecture sensible au contexte. De ce fait, le système de gestion de contexte doit gérer, d'une part, l'hétérogénéité de l'environnement d'exécution, et assurer, d'autre part, l'acceptation de cette adaptation par les utilisateurs. Plusieurs architectures ont été proposées

dans la littérature : *Context Toolkit* (Salber et al., 1999), *GAIA* (Ranganathan et Campbell, 2003), *Contexteur* (Rey et Coutaz, 2004), *COSMOS* (Conan et al., 2007) (Rouvoy et al., 2008), *pluggable middleware architecture* (Paspallis, 2009). Plusieurs Systèmes Sensibles au Contexte se basent sur ces architectures complexes, lesquelles intègrent des sous composants responsables de la représentation, de la gestion, du raisonnement et de l'analyse des informations contextuelles. Même si ces architectures différent entre elles, nous constatons que la plupart d'entre elles se composent de trois étapes essentielles : (i) l'acquisition ; (ii) la modélisation ; et (iii) la manipulation des informations contextuelles.

La première étape représente la phase d'*acquisition de contexte*. En se basant sur des capteurs physiques et logiques, le système est capable de capturer des informations contextuelles de nature hétérogène. Après la collecte des informations contextuelles, le système se charge de modéliser ces informations, de les traiter et éventuellement de les stocker. Cette deuxième étape représente la *modélisation des informations contextuelles*, dans laquelle le bon choix d'un modèle de contexte est un élément clé. Finalement, le système contrôle le niveau d'abstraction des données de contexte en interprétant les données de contexte brutes et en les transformant en information contextuelle de plus haut niveau d'abstraction. Ceci illustre la troisième étape du processus de gestion de contexte, laquelle représente la *manipulation des informations de contexte* en se basant sur des mécanismes de transformation et de raisonnement sur les données collectées.

2.3.3.1. Acquisition des informations contextuelles

L'acquisition des informations contextuelles repose sur la capacité d'un système à capturer des informations de l'environnement et à détecter les changements effectués. En d'autres termes, elle consiste à acquérir de l'environnement les informations jugées pertinentes en fonction des besoins du système. Ceci se base sur un module d'acquisition de contexte qui (i) capture directement les informations de l'environnement intégrant l'utilisateur et le système au travers de différentes sources hétérogènes, et (ii) les envoie à la couche supérieure pour différents usages. Les sources de contexte représentent les différents capteurs utilisés. Un capteur est une source matérielle ou logicielle qui peut générer un certain type d'information contextuelle. On distingue trois types de capteurs : les *capteurs physiques* directement installés dans l'environnement, les *capteurs virtuels* qui fournissent des informations contextuelles à partir d'applications ou de services logiciels, et les *capteurs logiques* qui utilisent plusieurs sources d'information contextuelles pour définir une autre information (Indulska et Sutton, 2003). Plus en détails, ces capteurs sont définis comme suit :

- **Le capteur physique** représente un dispositif matériel capable de fournir des données contextuelles de l'environnement autour de l'utilisateur et du système lui-même. Ceci peut se faire par le biais de différentes technologies. Par exemple, un capteur *photodiode* permet de capturer l'information sur la *lumière*. La *localisation* d'un utilisateur peut être capturée par la technologie *GPS* (*Global Positionning System*), laquelle permet de le localiser en utilisant un récepteur approprié ;

- **Le capteur virtuel** correspond à une application et à un service logiciel capable de fournir des informations sur le contexte. Par exemple, il est possible de capturer la tâche actuelle d'un employé en consultant son planning numérique. Ces capteurs, qui sont basés sur des composants logiciels, sont généralement moins coûteux que les capteurs physiques ;
- **Le capteur logique** repose sur plusieurs types d'information pour proposer une information dérivée de haut niveau. Selon Chaari (Chaari, 2007), ce type de capteur peut réutiliser des capteurs physiques et virtuels pour fournir un contexte de plus haut niveau, tel que les caractéristiques sociales, économiques, les compétences, etc. Cette information, dans certains travaux, peut être acquise à travers des méthodes de raisonnement. Par exemple, le système peut analyser l'historique d'un utilisateur afin de mettre à jour ses préférences.

Après avoir collecté les données contextuelles, ces informations doivent être représentées selon une structure bien déterminée, que nous présentons dans la section suivante.

2.3.3.2. Modélisation des informations contextuelles

L'étape de modélisation des informations contextuelles organise et modélise les informations de contexte capturées selon le modèle de contexte choisi et peut, par la suite, les stocker dans un répertoire. Un tel répertoire de contexte permet d'assurer la persistance des informations de contexte. Pour pouvoir stocker et transmettre au reste du système les informations contextuelles, il est impératif de définir un modèle de contexte pour les décrire. Ainsi, un modèle de contexte est requis pour pouvoir l'utiliser dans le système. En d'autres termes, le choix du modèle de contexte est l'élément clé de cette étape qui peut jouer un rôle central dans le traitement et l'interprétation de ces informations. La modélisation des informations contextuelles est présentée plus en détails dans la section 2.3.2.

2.3.3.3. Manipulation des informations contextuelles

Les données de contexte capturées lors de la première étape sont souvent des données brutes. Les capteurs interrogés remontent le plus souvent des données techniques qui ne sont pas appropriées à être utilisées par le concepteur du système (Baldauf et al., 2007). L'étape de *manipulation* consiste à traiter ces données afin de déduire et d'inférer des données de plus haut niveau. Les capteurs collectent les données de contexte de l'utilisateur et du système lui même. Ces données représentent des informations contextuelles qui sont *directes, explicites* et de *bas niveau* (*e.g.* la *localisation* de l'utilisateur est exprimée sous forme de *longitude* et *latitude*). Le manipulateur de contexte (ou interpréteur de contexte) est responsable du raisonnement et de l'interprétation de ces données et d'en déduire/inférer des informations contextuelles qui sont *implicites, indirectes* et de plus *haut niveau*. Ces informations contextuelles de haut niveau rajoutent de la connaissance sur le contexte acquis.

L'interprétation des données de contexte de bas niveau peut être faite par plusieurs opérations comme : opérations de calcul et de transformation, opérations de raisonnement, etc. Nous pouvons séparer ces opérations en deux catégories distinctes :

- **La transformation de contexte** : ce type d'opération permet de modifier la représentation de certaines informations de contexte en se basant sur des méthodes de classification, des calculs mathématiques, etc. Par exemple, la représentation d'une localisation exprimée sous forme de longitude et latitude peut être transformée en représentation sous forme d'adresse postale, plus significative, en utilisant des méthodes de transformation (Preuveneers, 2009). D'autres méthodes de transformation peuvent se baser sur des méthodes de calcul mathématique, telle que la transformation d'une représentation de la température exprimée en degré Celsius C° en une représentation exprimée en Fahrenheit F°.
- **Le raisonnement sur le contexte** : cette opération permet de dériver de nouvelles connaissances sur le contexte en se basant sur des faits existants et sur des règles de dérivations (Preuveneers, 2009). Le raisonnement sur le contexte peut combiner plusieurs règles de dérivation permettant de raisonner sur plusieurs informations contextuelles afin d'en déduire d'autres plus expressives ou plus précises par rapport au système. Dans ce sens, il est possible, par exemple, de déterminer l'activité courante de l'utilisateur en combinant plusieurs informations contextuelles, telle que le planning numérique de l'utilisateur, sa localisation, etc.

Finalement, face à la grande variété des modèles de contexte que nous avons détaillés dans la section 2.3.2 et aux différents moyens de capture et de gestion de contexte, nous proposons dans la section suivante, un cadre permettant d'analyser ces différents modèles de contexte.

2.3.4. Cadre d'analyse et de comparaison des modèles existants

Selon Mostéfaoui et al. (Mostefaoui et al., 2004), la pertinence des informations contextuelles diffère d'un domaine à un autre selon leur utilisation. Cette affirmation peut être observée dans les différents travaux de modélisation de contexte qui ont été présentés dans la section 2.3.2, dans lesquels divers éléments de contexte (profil de l'utilisateur, préférences de l'utilisateur, localisation, dispositif, etc.) sont observés pour différentes raisons (découverte et configuration de services, personnalisation et adaptation du contenu, etc.). L'immense diversité présente par les modèles de contexte, autant en termes d'éléments de contexte observés, que de formalismes utilisés, rend souvent difficile leur évaluation. Nous proposons ainsi, dans le cadre de cette thèse, un cadre qui analyse et compare les différents modèles de contexte. Ce cadre a comme objectif d'aider à la compréhension et à l'analyse de ces modèles. Il permet d'étudier et de classifier les modèles de contexte, selon différents critères, dans la perspective d'aider le concepteur à choisir le modèle qui lui convient ou de le guider à en déduire un nouveau modèle à partir de l'existant.

2.3.4.1. Les critères d'analyse des modèles de contexte

Les différents critères d'évaluation que nous proposons sont utilisés pour l'analyse des modèles de contexte. Ces critères sont choisis afin de mettre en avant les caractéristiques de chaque modélisation, aidant ainsi l'utilisateur à faire son choix. Ces critères sont :

- **Information** : *Quelles sont les informations de contexte qui doivent être observées ?* Ce critère permet d'identifier les informations les plus pertinentes qui vont être observées et ensuite utilisées par une application particulière. Ces informations forment le contexte qui doit être modélisé ;
- **Action** : *Quelles sont les actions prises par le système sur la base des informations contenues dans un modèle de contexte ?* Ce critère représente l'ensemble d'actions qui peuvent être entreprises par le système s'appuyant sur le modèle de contexte. Il montre comment le modèle de contexte est utilisé ;
- **Structure (Str)** : *Comment cette information est-elle représentée ?* Ce critère représente la structuration interne de l'information contextuelle. Il permet de mesurer le degré de formalisation des informations contextuelles ;
- **Capture (Capt)** : *Quelle est la méthode utilisée pour capturer l'information contextuelle ?* Pour chaque application, afin d'obtenir des informations sur le contexte d'utilisation, il faut avoir une stratégie de capture qui détecte les informations contextuelles et notifie le reste du système lorsqu'il y a un changement ;
- **Maintenance (Maint)** : *Comment pouvons-nous mettre à jour cette information et garder sa pertinence ?* L'objectif de ce critère est de préciser les techniques utilisées pour la mise à jour de l'information contextuelle et de définir les stratégies utilisées pour cela ;
- **Raisonnement (Rais)** : *Est-ce qu'il existe un moteur de raisonnement capable d'interpréter les informations de contexte ?* Ce critère permet de vérifier si des techniques sont mises en place afin de dériver de la connaissance contextuelle de haut niveau à partir des données de contexte de bas niveau ;
- **Expressivité (Exp)** : *Est-ce que le modèle peut représenter des informations complexes ?* Ce critère mesure la capacité de représenter des entités et des relations *complexes*. Ceci va permettre d'améliorer l'interopérabilité qui demande une compréhension commune de l'information obtenue. Ce critère est généralement en conflit mutuel avec l'efficacité du raisonnement ;
- **Efficacité (Eff)** : *Est-ce qu'il y a un équilibre entre l'expressivité du modèle de contexte et la complexité du traitement mis en place ?* Ce critère permet de mesurer la performance d'un type de modèle de contexte en termes de quantité d'informations qui doit être traitée et de complexité du modèle de contexte lui-même ;
- **Ambiguïté et Incomplétude (Ambig)** : *Est-ce que les informations de contexte capturées sont de bonne qualité ?* Ce critère permet de déterminer si les informations contextuelles capturées sont incomplètes, ambiguës ou incertaines. Il permet de mesurer la qualité et la pertinence des informations de contexte ;

- **Effort de Programmation (Prog)** : *Est-ce que les développeurs ont fourni un effort remarquable pour la définition et l'utilisation d'un modèle de contexte particulier au sein de l'application ?* Ce critère reflète l'effort dépensé par les développeurs afin de mettre en place leur modèle de contexte.

2.3.4.2. Analyse comparative des modèles de contexte

Les modèles de contexte que nous avons étudiés dans la section 2.3.2 ont d'abord été classifiés selon leur approche de modélisation, avant d'être analysés selon les critères précédemment énoncés. Un aperçu de cette analyse comparative est illustré au Tableau 1.

Pour les deux critères *information* et *action*, nous avons décrit textuellement les informations contextuelles utilisées et les actions entreprises faisant appel au modèle de contexte pour chaque approche. Les autres critères sont évalués par un indicateur d'évaluation qualitatif : (++) score élevé ; (+) score acceptable ; (-) score médiocre. Par exemple, le critère de *structure* indique le degré de formalisation du modèle de contexte. Ainsi, nous indiquons (++) si l'information est très structurée, comme dans les modèles basés sur les ontologies ; (+) si elle est semi-structurée, généralement XML ou autres modèles orientés objets ; ou (-) si elle n'est pas structurée, comme dans les modèles clé-valeur.

Le critère de *capture* se réfère au degré d'automatisation du processus d'acquisition de contexte. Nous indiquons (++) si l'information est capturée périodiquement ; (+) si elle répond aux changements de contexte suite à une approche basée sur les événements ; ou (-) si elle est acquise manuellement. Le critère de *maintenance* évalue la stratégie utilisée pour mettre à jour les informations de contexte. Nous indiquons (++) si l'information est maintenue automatiquement ; (+) si elle est semi-automatique (sur demande) ; ou (-) si elle est maintenue manuellement. Le critère de *raisonnement* montre une évaluation des techniques de raisonnement. Nous désignons par (++) le fait qu'elle dispose d'un moteur de raisonnement ; (+) qu'elle ait un mécanisme de raisonnement ad-hoc lequel est codé en dur ; ou (-) qu'elle dispose d'un mécanisme de raisonnement assez faible.

De même, les critères *d'expressivité, d'efficacité, d'ambiguïté et d'effort de programmation* sont évalués de la même façon. Ils sont évalués (++) pour les modèles qui sont les plus expressifs et assurant un niveau élevé d'interopérabilité, qui sont les plus efficaces, qui offrent le moyen de détecter et de gérer l'ambiguïté et l'incomplétude des informations contextuelles, et qui ne demandent pas beaucoup d'effort de programmation pour les réutiliser. Ils sont ainsi évalués (-) pour les modèles les moins expressifs, les moins efficaces, qui ne gèrent pas la qualité des données de contexte et qui demandent un effort remarquable de programmation.

Tableau 1. Cadre d'analyse et de comparaison des modèles de contexte (Najar et al., 2009)

Approche de modélisation de Contexte	Information	Action	Str	Capt	Maint	Rais	Exp	Eff	Ambig	Prog
Approche **Schilit** *(Schilit et al., 1994)*	- Localisation - Identité des personnes et objets à proximité	- Sélection approximative - Reconfiguration automatique de contexte	-	++	?	-	-	++	-	++
Approche **CC/PP** *(Lemlouma, 2004)*	- Utilisateur - Terminal	- Adaptation de la présentation du contenu pour les dispositifs mobiles	+	-	-	-	+	+	-	+
Approche **Kirsch-Pinheiro** *(Kirsch-Pinheiro, 2006)*	- Info physique de l'utilisateur -Info collaborative de l'utilisateur	- Adaptation du contenu d'un utilisateur mobile dans un groupe	+	-	-	+	+	+	+	++
Approche **CML** *(Henricksen et al., 2002)* *(Henricksen et Indulska, 2006)*	- Préférence utilisateur - Situation utilisateur	- Prise de décision dépendante du contexte et des préférences - Invocation automatique des actions selon changement de contexte	+	+	-	-	++	-	+	+
Approche **MUSIC** (Reichle et al., 2008)	- Utilisateur - Dispositif - Environnement - Entité de calcul	- Découverte de services - Composition de services	++	++	++	+	++	-	++	+

Le modèle de contexte proposé par Schilit et al. (Schilit et al., 1994), par exemple, est structuré selon un formalisme *clé-valeur,* présentant une *structuration assez simple*, laquelle ne permet pas des structures complexes (-). Les informations de contexte incluent la *localisation et l'identité des personnes et des objets à proximité*. Le système déploie différentes technologies pour *capturer périodiquement* la localisation des objets mobiles, le plus souvent la localisation des personnes au voisinage (++). Ce modèle de contexte ne se base que sur des mises en correspondance *(matching)* exactes entre les valeurs capturées, représentant ainsi un modèle de contexte avec un niveau de raisonnement très faible (-), mais qui gagne en terme d'efficacité (++). De plus, l'interprétation dans les modèles de contexte utilisant une structure simple de clé-valeur dépend de l'application, ce qui signifie que l'interopérabilité devient un problème si les clés et les valeurs ne sont pas standardisées, affectant ainsi l'expressivité de ce modèle (-). En outre, la simplicité des paires clé-valeur ne permet pas de gérer l'ambiguïté des informations contextuelles (-), mais présente une certaine facilité en terme d'effort de programmation (++).

Nous analysons de même les travaux de Lemlouma (Lemlouma, 2004) qui propose un modèle de contexte se basant sur la structure CC/PP (+) pour décrire sous forme de profil les préférences de l'utilisateurs et les capacités physiques et logicielles d'un terminal. Cet auteur utilise le modèle CC/PP basé sur RDF permettant ainsi d'améliorer le niveau d'expressivité de ce modèle (+) et d'améliorer l'interopérabilité entre les différents terminaux et ressources. De plus, le modèle proposé repose sur un mécanisme manuel de capture (-) dans lequel le dispositif client donne, au départ, une description de ses caractéristiques qui peut être, par la suite complétée, par le système. Néanmoins, les informations contextuelles ou ses changements ne sont pas capturées automatiquement. Les profils contenant les informations de contexte sont mis à jour régulièrement, mais manuellement, afin d'intégrer de nouveaux terminaux et logiciels ou pour changer les préférences des utilisateurs (-). En outre, dans ses travaux, Lemlouma (Lemlouma, 2004) ne supporte pas de mécanismes de raisonnement et d'interprétation (-). De plus, cet auteur traite l'ambiguïté et l'incomplétude des informations contextuelles exclusivement au niveau de l'application (+).

L'approche proposée par Kirsch-Pinheiro et al. (Kirsch-Pinheiro et al., 2004), quant à elle, repose sur une structure de modélisation orientée objets (+) pour la description des éléments de contexte, telle que les informations physiques (la localisation de l'utilisateur, le dispositif, etc.) et collaboratives (rôle de l'utilisateur, activités, etc.), dans le but de personnaliser le contenu fourni par des systèmes collaboratifs basés sur le Web. Néanmoins, la capture et la maintenance de ces éléments de contexte sont considérées comme extérieures au modèle et ne sont pas prises en compte (-). Par ailleurs, ces auteurs proposent également une méthode de raisonnement *ad-hoc* (+), à travers un ensemble de mesures de similarité permettant de comparer les éléments de contexte et leurs relations. Comparée à d'autres modèles, cette approche gagne en expressivité (+), avec cette nouvelle modélisation orientée objets, mais demeure peu adaptée au partage des connaissances dans un environnement ouvert et dynamique. De plus, elle nécessite certaines normes d'exécution de bas niveau entre les applications pour assurer l'interopérabilité. Enfin, ce modèle de contexte offre la possibilité de représenter une information contextuelle incomplète ou même ambiguë (+).

Nous continuons cette étude comparative par l'analyse du modèle de contexte CML, proposé par (Henricksen et al., 2002), lequel profite de la structuration de la modélisation graphique (+). Henricksen et al. (Henricksen et al., 2002) gèrent la capture de contexte de diverses sources en mettant en place un composant de contexte qui collecte les informations dès qu'il y a un changement (+). De plus cette modélisation permet l'interprétation des informations contextuelles, à travers de conditions simples, notamment l'évaluation des assertions ainsi que des requêtes SQL. Pour traiter des conditions plus complexes, Henricksen et al. (Henricksen et al., 2002) définit une grammaire pour la formulation des situations qui sont exprimées en utilisant la logique des prédicats (+). Par ailleurs, l'implémentation de ce modèle conduit à un code plus propre qu'un modèle non structuré (+), mais à des applications plus difficiles à maintenir (-). En outre, CML fournit également un soutien plus complet à la saisie et à l'évaluation de l'information imparfaite et ambiguë que d'autres approches de modélisation de contexte (+).

Reichle et al. (Reichle et al., 2008) ont contribué avec la proposition du modèle de contexte utilisé dans le projet IST-MUSIC (IST-MUSIC, 2010). Cette ontologie pour assurer l'adaptation de systèmes à base de composants OSGI, en prenant en considération les éléments de contexte énumérés. Le modèle, basé sur une ontologie, est extensible, bien structuré et facile à comprendre (++), composé de deux niveaux hiérarchiques, afin de faciliter sa réutilisation dans divers domaines. L'acquisition de contexte sur la plateforme MUSIC repose sur des plug-ins de contexte (Paspallis, 2009) agissant comme des fournisseurs de contexte qui capturent dynamiquement des informations contextuelles (++) et les modélisent sémantiquement. Ces plug-ins de contexte assurent aussi l'interprétation et le raisonnement sur les informations contextuelles, afin de dériver de la connaissance de plus haut niveau, en se basant sur d'autres plug-ins (+). De plus cette ontologie se caractérise principalement par son expressivité et son interopérabilité facilitant ainsi son usage et ses modifications (++) mais impactant ainsi son efficacité en terme de complexité de traitement et de quantité d'informations à traiter (-).

A la fin de cette analyse comparative des différents modèles de contexte, nous concluons que, même si d'autres modèles pourraient aussi être adaptés à certaines applications, les approches basées sur l'ontologie semblent en général être les plus prometteuses en termes de raisonnement et d'expressivité, mais l'impact sur l'efficacité et la complexité de programmation sont les deux principaux obstacles à franchir.

Dans un environnement pervasif, la notion de contexte, que nous avons explorée dans les sections précédentes, représente les informations capables d'être observées dans cet environnement. Ainsi, les Systèmes d'Information intégrés dans un tel environnement pervasif se doivent de prendre en considération cette notion de contexte (*cf.* section 2.3.1) afin de s'adapter à la dynamique de l'environnement. En d'autres termes, ils se doivent d'être *sensibles au contexte* (*cf.* section 2.2.2). Cette nouvelle classe de SI, qui s'intègre dans cet environnement pervasif, est fondée sur cette notion de contexte et de sensibilité au contexte afin d'assurer, d'une part, la *gestion de l'hétérogénéité et de la dynamique de l'environnement*, et d'autre part, la *transparence de ce système*. De plus, afin de représenter

les informations contextuelles capturées et de s'adapter en conséquence, ces systèmes doivent définir au préalable un modèle de contexte, comme nous l'avons discuté dans la section 2.3.2.

Dans la section suivante, nous présentons cette nouvelle vision des SI, présentée sous le nom de *Systèmes d'Information Pervasifs* (SIP).

2.4. Les Systemes d'Information Pervasifs et caracteristiques

Au sein des organisations, les Systèmes d'Information (SI) sont directement impactés par l'arrivée de l'Informatique Pervasive. La mobilité qu'apportent ces nouvelles technologies a étendu les SI bien au-delà des frontières physiques de l'organisation. De ce fait, émerge une nouvelle classe de Systèmes d'Information, les *Systèmes d'Information Pervasifs* (SIP) (Birnbaum, 1997). Cette nouvelle génération de SI est le fruit d'un important changement dans la manière dont nous travaillons et dont les technologies nous supportent dans notre quotidien. Nous passons d'un modèle statique, dans lequel les travailleurs n'interagissent avec un processus métier que durant leur « temps de travail » et dans des circonstances bien définies (*e.g.* assignés à leurs ordinateurs de bureau), à un modèle dynamique, assuré par l'évolution des réseaux sans fil et des dispositifs mobiles, dans lequel les travailleurs se caractérisent par leur mobilité.

Les Systèmes d'Information Pervasifs (SIP) représentent une classe émergente de SI dans laquelle les technologies de l'information (IT) sont graduellement intégrées à l'environnement physique (Kourouthanassis et Giaglis, 2006). A l'inverse des SI traditionnels, les Systèmes d'Information Pervasifs s'intègrent progressivement à l'environnement physique, passent à l'arrière-plan, gardent une trace des activités des utilisateurs, analysent les informations et interviennent lorsque cela est nécessaire, afin de mieux répondre aux besoins de ces utilisateurs (Kourouthanassis et Giaglis, 2006). Il s'agit d'intégrer l'informatique dans l'environnement physique de manière invisible, telle que soutenue depuis plusieurs années par l'Informatique Pervasive (Weiser, 1991).

Les Systèmes d'Information Pervasifs se veulent alors une réponse à cette importante évolution des SI. Selon (Kourouthanassis et Giaglis, 2006), les SIP revisitent la manière dont nous interagissons avec les SI : le paradigme d'interaction passe du simple bureau classique et complétement maîtrisé à un ensemble de dispositifs multiples, très hétérogènes et intégrés dans un environnement hautement dynamique. Toujours selon ces auteurs, les SIP se caractérisent par l'hétérogénéité des terminaux qui permettent d'accéder à leurs services. Ces terminaux peuvent être eux-mêmes mobiles ou intégrés directement à l'environnement. A l'opposé des SI traditionnels, dans lesquels les dispositifs d'accès sont stationnaires, les SIP supportent des dispositifs mobiles qui peuvent être transportés partout par les utilisateurs. Néanmoins, les SIP doivent ainsi gérer cette hétérogénéité et veiller à la bonne interaction entre l'utilisateur et l'environnement physique (Kourouthanassis et Giaglis, 2006). En d'autres termes, les SIP revoient la façon dont nous interagissons avec les ordinateurs par l'introduction de nouvelles modalités d'accès à ces systèmes. Selon (Kitsios et al., 2010), les

SIP ont pris du recul par rapport aux interactions originales d'un utilisateur avec son SI et ils considèrent les dispositifs comme entièrement intégrés, d'une manière transparente, à l'environnement physique. Avec cette évolution, le SI traditionnel de bureau (*DSI - Desktop Information System*) est considéré alors comme un des dispositifs d'accès mis à disposition de leurs utilisateurs. Par conséquent, des interactions continues sont rendues possibles par ces dispositifs, qu'ils soient mobiles ou intégrés dans l'environnement physique (Kourouthanassis et Giaglis, 2006). En découle ainsi la possibilité d'offrir aux utilisateurs des nouveaux services innovants, mais également un système qui serait capable de percevoir son contexte d'utilisation, de gérer la mobilité des utilisateurs et des services, afin de mieux s'accommoder aux besoins et aux désirs de ces utilisateurs.

Les SIP s'opposent aux SI traditionnels en s'intégrant de plus en plus dans le monde physique. Les SI traditionnels intègrent l'intelligence uniquement à l'intérieur du système. Dans le cas des SIP, l'intelligence du système ne réside plus uniquement dans l'ordinateur, mais elle est aussi intégrée dans le monde physique (Kourouthanassis et al., 2008).

Au-delà de l'hétérogénéité, l'environnement dans le quel s'intègre les SIP se caractérise également par sa dynamique. Selon (Kourouthanassis et Giaglis, 2006), les SIP s'inscrivent dans un environnement particulièrement dynamique, composé d'une multitude d'artefacts capables de percevoir le contexte de l'utilisateur et de gérer sa mobilité. Il s'agit, par opposition aux environnements de *bureau* qui caractérisent traditionnellement les SI, d'un environnement hautement dynamique, dont l'état varie en fonction non seulement des actions et de la mobilité de leurs utilisateurs, mais également de l'état de ses éléments, eux-mêmes très variables. Selon (Hagras, 2011), la nature dynamique des environnements pervasifs leur impose une capacité d'adaptation à des conditions d'opération changeantes et à des utilisateurs dont les préférences et le comportement sont également variables. Quelques soient les variations auxquelles il est exposé, un système opérant sur un tel environnement se doit d'être adaptable. En d'autres termes, un système dit pervasif doit être capable d'accomplir les fonctionnalités sollicitées, malgré les changements dans les conditions environnantes ou dans l'état du système (Römer et Friedemann, 2010).

Ainsi, la dynamique des environnements pervasifs apporte aux SIP une autre caractéristique importante : la *sensibilité au contexte* (*cf.* section 2.2.2). Plusieurs auteurs (Vanrompay et al., 2011) (Baldauf et al., 2007) soulignent le rôle central de la sensibilité au contexte dans les systèmes pervasifs. Pour (Kourouthanassis et al., 2008), contrairement aux SI traditionnels, dans lesquels une réponse du système est précédée d'une action de l'utilisateur, les SIP doivent être proactifs, réagissant aux stimuli de l'environnement. Ils doivent s'adapter aux préférences de l'utilisateur et au contexte d'utilisation.

A l'opposé, ces nouveaux Systèmes d'Information Pervasifs se distinguent également des applications et des plateformes sensibles au contexte, traditionnellement proposées dans le domaine de l'Informatique Pervasive (Baldauf et al., 2007) (Preuveneers et al., 2009) (Geihs et al., 2009). Nous discuterons ces applications dans le prochain chapitre (*cf.* chapitre 3) qui s'intéresse aux systèmes orientés services sensibles au contexte. Contrairement à ces derniers,

dans lesquels le caractère dynamique et l'autogestion sont des caractéristiques essentielles, les SIP se doivent de rester maîtrisables et maîtrisés. En réalité, cette sensibilité au contexte ne doit pas se faire au dépend de la transparence. Selon (Dey, 2011), lorsque les utilisateurs ont des difficultés à former un modèle mental de l'application, ils ont moins envie de l'adopter et de l'utiliser. Malgré les capacités d'adaptation au contexte et un comportement proactif, les SIP doivent rester *compréhensibles* à leurs utilisateurs, d'autant plus qu'il s'agit, avant tout, des Systèmes d'Information. Les utilisateurs doivent garder leur confiance en ces systèmes qui sont là pour répondre à leurs besoins.

Les SIP doivent ainsi faire coexister deux mondes aux antipodes. Ils se doivent d'être sensibles au contexte, de prendre en compte le caractère dynamique des environnements pervasifs, sans pour autant perdre complétement le caractère maîtrisé et prédictible propre aux SI. En tant que Systèmes d'Information, les SIP doivent être conçus afin de mieux *satisfaire les besoins* de leurs utilisateurs. Leur objectif est de rendre ces utilisateurs plus performants par une meilleure prise en compte de l'environnement pervasif dans lequel ils se trouvent. Il s'agit de répondre aux besoins des utilisateurs de manière adaptée, quelques soient les conditions d'usage. La satisfaction de l'utilisateur est une priorité pour les SIP. Contrairement aux SI traditionnels, dans lesquels c'était à l'utilisateur de s'adapter au système, les SIP doivent désormais prendre en compte l'environnement et le contexte d'utilisation afin de mieux s'adapter et de fournir aux utilisateurs le service qui correspond au mieux à ses besoins et à son contexte. Cependant, cette adaptation ne doit pas se faire à n'importe quel prix. Même s'il doit tirer profit de l'environnement dynamique et des opportunités qu'un tel environnement peut lui offrir, le comportement d'un SIP, avec les services et les fonctionnalités qu'il offre à ses utilisateurs, doit rester *prédictible*, afin d'assurer la gouvernance de ces systèmes et la confiance des utilisateurs en eux.

De plus, certains auteurs, tels que Kourouthanassis et al. (Kourouthanassis et al., 2008), considèrent ainsi le degré d'omniprésence *(pervasiveness)* d'un SIP donné. Ces auteurs définissent l'*omniprésence* comme *la mesure dans laquelle un Système d'Information est constitué d'artefacts technologiques interconnectées, diffusés dans l'environnement qui les entoure, qui travaillent ensemble pour soutenir de façon ubiquitaire les tâches et les objectifs de l'utilisateur à un niveau, organisationnel, de groupe, ou individuel d'une manière sensible au contexte.* En se basant sur cette définition, les SIP s'organisent selon trois dimensions principales : (i) **la sensibilité au contexte** représente la capacité à percevoir l'information contextuelle en observant l'utilisateur, le système et l'environnement dans le but d'adapter ses fonctionnalités d'une façon dynamique et proactive ; (ii) **l'ubiquité** représente la capacité à fournir aux utilisateurs un accès continu aux ressources d'information indépendamment de leur localisation à l'intérieur des limites du système ; et (iii) **la diffusion** représente la capacité d'un système à intégrer des éléments invisibles à l'utilisation dans l'environnement physique, et dont l'interaction se fait par le biais d'interfaces physiques.

Afin d'assurer ces trois dimensions, les Systèmes d'Information Pervasifs (SIP) doivent faire face à un ensemble de challenges, dont :

- ***l'adaptabilité*** : les SIP doivent s'adapter aux changements de l'environnement afin de gérer sa dynamique. S'ils souhaitent augmenter leur utilisabilité et leur efficacité, ces systèmes sont amenés à adapter leur fonctionnement par la prise en compte du contexte environnant. Comme l'environnement d'exécution change en raison de la mobilité de l'utilisateur et du dynamisme de l'environnement, le système doit changer son comportement pour prendre en considération cette mobilité et ce dynamisme. Ainsi, les SIP doivent désormais s'adapter à l'environnement et au contexte de l'utilisateur en toute transparence afin de lui offrir les services les plus appropriés ;
- ***l'interopérabilité*** : un environnement pervasif intègre divers types de dispositifs informatiques, de réseaux et d'entités logicielles et matérielles. Tous ces éléments se caractérisent par leur hétérogénéité. Ainsi, en raison du grand nombre de services hétérogènes offerts par l'environnement pervasif, l'*interopérabilité* est devenue une nécessité à tous les niveaux pour les SIP. Par exemple, la découverte et la composition de services au sein de ces systèmes doivent pouvoir tirer profit des fonctionnalités et des services de différentes natures, et pour cela ils doivent être interopérables ;
- ***la sécurité*** : dans le cadre d'un SI, les mécanismes de sécurité représentent un défi important. Un SIP demeure un système fermé, qui nécessite un certain niveau de maîtrise et de contrôle. Toutefois, l'immersion des fonctionnalités d'un SI dans l'environnement pervasif expose celui-ci à un risque de perdre le contrôle et la maitrise du SI dans un tel environnement hétérogène et hautement dynamique. En effet, un comportement totalement dynamique, avec une prise en compte opportuniste des ressources et des services disponibles dans l'environnement et l'autogestion des services offerts, peut être vu comme une menace pour le SI. Le défi ici est de fournir les mécanismes de sécurité adéquats pour contrôler intelligemment l'accès aux ressources informatiques et aux services offerts par le SIP, sans pour autant perdre la confiance des utilisateurs.

Toutefois, nous observons à travers la littérature, que cette nouvelle notion de SIP demeure une vision conceptuellement étudiée mais qui n'est pas mise en place par de vrais formalismes de conception, ni d'architecture pour les réaliser. Ceci ouvre de nouvelles opportunités pour traiter les SIP et les mettre en place dans un environnement pervasif et de les concevoir de telle sorte qu'ils soient acceptés par les DSI.

2.5. Conclusion

Dans le cadre de ce premier chapitre de l'état de l'art, nous avons introduit l'émergence de l'Informatique Pervasive. Celle-ci a pour but de rendre accessible toutes sortes de services, n'importe où et à tout moment. Cette volonté d'affranchir l'utilisateur des contraintes actuelles d'utilisation d'un ordinateur lui rend sa liberté d'actions, notamment sa liberté de mobilité.

Au cœur de l'Informatique Pervasive se trouve la notion de *sensibilité au contexte*. Cette notion représente la prise en compte du contexte d'utilisation lors du développement

d'applications qui fournissent, en conséquence, des solutions mieux adaptées à leur usage. Nous avons signalé, que dans ce cadre, le contexte représente un élément clé dans les mécanismes d'adaptation prônés par ces systèmes dits *sensibles au contexte*. A travers les différentes définitions discutées dans ce chapitre, nous avons souligné que la notion de contexte est une notion très large et vague, qui peut varier selon le domaine, par exemple. Ainsi, face à ces différentes définitions de contexte, se pose la question de la modélisation de *contexte afin de pouvoir le représenter et le limiter*. Différents modèles de contexte ont été présentés allant des plus simples modèles à paires clés-valeurs, aux plus expressifs modèles basés sur les ontologies. Nous avons ainsi constaté que les ontologies sont de plus en plus utilisées. Ceci est principalement dû aux propriétés formelles et à l'expressivité des ontologies, ainsi qu'aux moteurs d'inférence associés. Les ontologies permettent également d'assurer l'interopérabilité souhaitée au niveau sémantique, et donc la réutilisation de ces modèles avec une certaine cohérence sémantique. Ceci facilite donc la gestion de contexte.

Ensuite, nous observons que cette Informatique Pervasive a impacté l'usage des Systèmes d'Information (SI) existants et a fait apparaître une nouvelle génération de SI, nommé les *Systèmes d'Information Pervasifs*. En effet, la dernière décennie a été remarquablement marquée par le changement dans la manière dont nous travaillons. Les différents travaux de recherches menés dans cette thématique ont montré que la mobilité qu'apportent les nouvelles technologies a étendu les SI bien au-delà des frontières physiques de l'organisation. Par la prise en compte de la notion de *contexte* et de *sensibilité au contexte*, les SIP répondent à la problématique d'adaptation à l'environnement physique dans lequel ils s'intègrent et qui se caractérise par sa haute dynamique et son changement fréquent. Toutefois, ces SIP doivent faire face à d'autres problématiques à savoir la gestion de l'hétérogénéité et la compréhension des besoins des utilisateurs afin d'assurer une certaine transparence nécessaire dans le cadre des SIP. Nous abordons ces sujets dans le chapitre suivant de l'état de l'art.

Chapitre 3. Systemes d'Information Pervasifs et l'orientation service

3.1. Introduction

Depuis des années, la notion de *service* a été largement étudiée dans la littérature. Cette notion peut être vue, d'une manière générale, comme étant des composants logiciels qui sont désignés de telle sorte qu'ils supportent des demandes métiers évolutives. Cette notion de service n'a cessé d'évoluer avec l'apparition des services Web et du Web sémantique permettant de garantir un meilleur niveau d'interopérabilité, d'intégration, de compréhension et d'automatisation. Cette évolution a encouragé d'autres chercheurs à étudier la notion de services sous un angle plus proche de l'utilisateur. Ceci a fait apparaître les *services sensibles au contexte*, qui doivent s'adapter aux changements de l'environnement, et les *services intentionnels*, qui décrivent les intentions qu'ils peuvent satisfaire.

Dans le cadre des Systèmes d'Information Pervasifs (SIP), l'orientation service permet de répondre au besoin de gestion de l'hétérogénéité technique de l'environnement dans lequel évoluent ces systèmes et des actions que ces systèmes proposent afin de satisfaire les besoins des utilisateurs. En effet, l'indépendance des services par rapport aux aspects technologiques et à leurs implémentations est un des moyens permettant de masquer l'hétérogénéité technologique des environnements pervasifs.

Avec l'évolution des SI vers les SIP, de nouveaux challenges sont apparus ou ont pris une forme différente. En effet, afin de construire un SIP qui prend en considération à la fois le caractère dynamique des environnements pervasifs et le caractère contrôlé et maîtrisé des SI, il est important de répondre à certains défis, tels que l'adaptabilité, la découverte, la composition et la prédiction dynamique des services. Dans le cadre de cette thèse, nous soulevons la problématique de la découverte et de la prédiction de service dans le cadre d'un SIP transparent, proactif et centré utilisateur.

Ce chapitre présente un état de l'art sur l'orientation service. Nous commençons par présenter la notion de services et l'émergence des visions techniques, sémantiques et intentionnelles de cette notion. L'orientation service, selon nous, est un concept clé dans la conception des SIP qui doivent faire face à un ensemble de défis. Nous présentons, par la suite, les différents défis auxquels un SIP orienté service doit répondre. Nous nous focalisons sur deux d'entre eux : la découverte de services et la prédiction de services. Nous attribuons une attention particulière aux différentes approches de découverte de services apparues dans la littérature. Nous commençons par analyser quatre types d'approches, à savoir : (i) la *découverte de services sémantiques* ; (ii) la *découverte de services sensibles au contexte* ; (iii) la *découverte de services intentionnels ;* et (iv) la *découverte de services sensibles au contexte et intentionnels*. Par la suite, nous nous concentrons sur deux approches de prédiction de

services, à savoir : (i) la *prédiction de contexte ;* et (ii) la *recommandation de services* selon le contexte. Finalement, nous présentons un récapitulatif suite à cette analyse des approches de découverte et de prédiction de services.

3.2. LA NOTION DE SERVICE

A travers la littérature, la notion de *services* correspond à un concept largement répandu. Originairement, la notion de service est apparue avec l'émergence de l'informatique orientée services (SOC) (*Service Oriented Computing*) afin de gérer le problème d'interopérabilité entre des applications et des architectures hétérogènes (Papazoglou et Georgakopoulos, 2003) (Papazoglou, 2003). Le paradigme SOC utilise les *services* comme étant les éléments fondamentaux afin de supporter un développement rapide, fiable et à moindre coût d'applications logicielles distribuées dans des environnements hétérogènes avec une facilité de composition (Papazoglou et al., 2008).

La notion de service est employée afin de fournir des abstractions de haut niveau pour l'organisation d'applications à grande échelle sur des environnements ouverts. L'usage des services contribue à implémenter et à configurer des applications logicielles d'une manière qui améliore leur productivité et leur qualité (Huhns et Singh, 2005). Plus spécifiquement, ces *services* diffèrent des artefacts traditionnels par leur nature autonome, auto-descriptive, réutilisable et portable. Ils sont conçus comme un ensemble de modules logiciels autonomes qui peuvent être exposés, publiés, découverts, composés et négociés à la demande d'un client et invoqués par d'autres applications (Papazoglou et al., 2008).

Dans le paradigme SOC, la notion de *service* est matérialisée par un composant logiciel souvent associé à un ensemble de fonctionnalités particulières dont l'interface est clairement définie, voir standard, et dont le fonctionnement interne est inconnu des clients (Papazoglou et al., 2008). En d'autres termes, toujours selon ces auteurs, les services exécutent des fonctions qui peuvent être aussi bien de simples requêtes dans un formulaire, que des processus métiers complexes. En effet, les processus métiers sont supportés par des fonctionnalités qui sont implémentées comme des services. Ces services représentent des composants logiciels qui sont désignés de telle sorte qu'ils supportent les demandes métiers qui sont évolutives.

Dans la même lignée, Alonso et al. (Alonso et al., 2004) définissent les *services* comme *des éléments logiciels auto-décrits, indépendants de la plateforme et accessibles par une interface standard*. Une définition plus générale est donnée par Issarny et al. (Issarny et al., 2007), qui considèrent un *service* comme une *entité indépendante, dotée d'interfaces bien définies et pouvant être invoquée de manière standard, sans requérir de son client une quelconque connaissance sur la manière dont le service réalise réellement ses tâches*.

Quelle que soit la définition utilisée, il est important de souligner que les fonctionnalités (ou tâches) attachées aux services peuvent se trouver à des niveaux de granularité très différents. Même si, bien souvent, la notion de service fait référence aux Services Web, celle-

ci est, en réalité, bien plus large, allant d'une vision orientée technologie à une vision orientée « *business* » (les services offerts par une organisation, par un SI). Même dans une optique purement technologique, cette notion dépasse le cadre des Services Web et peut correspondre à des technologies variées telles que les *ESB* (Roshen, 2009) ou les composants *OSGi* (souvent vus comme de « micro-services ») (Hall et al., 2011).

Un autre aspect clé se dégage de ces multiples définitions : le faible couplage entre le client et le fournisseur de service. En effet, l'absence de dépendance entre les fournisseurs de services et leurs consommateurs permet d'ordonner les services dans de nombreux flux réalisant différents processus métiers (Fremantle et al., 2002) (Papazoglou et al., 2008). Le client n'a pas besoin de connaître la manière dont le service fonctionne ou est implémenté pour faire appel à ses fonctionnalités. C'est ce faible couplage qui rend la notion de service particulièrement attractive pour les environnements pervasifs, puisque ces environnements se caractérisent par la volatilité de leurs éléments (Vanrompay et al., 2011). La notion de service se comporte ainsi comme une « *boîte noire* » : rien de son contenu interne n'est visible aux clients, seule son interface l'est.

La notion de service joue un rôle clé dans l'architecture *SOA*, introduite dans la section 3.3, et devient l'unité informatique de base pour supporter le développement et la composition de services de plus en plus complexes, qui à leur tour peuvent être utilisés pour créer des applications flexibles et dynamiques.

3.3. L'ARCHITECTURE ORIENTEE SERVICES : SOA

Pour soutenir l'intégration des applications basées sur différents processus métiers, la modélisation de services est supportée par l'architecture orientée service (*SOA – Service Oriented Architecture*) (Papazoglou et al., 2008). L'orientation service s'est introduite à différents niveaux organisationnels au sein des entreprises. Elle s'appuie sur la technologie pour faire face à la demande croissante d'une plus grande *intégration*, *flexibilité* et *agilité* au sein de l'entreprise.

Depuis la fin des années 1990, de nombreuses définitions de l'architecture SOA ont été publiées (Allen et Frost, 1998) (Alonso et al., 2004). Keith et al. (Keith et al., 2006) considèrent SOA comme un *concept de modélisation métier et une architecture technique présentée sous la forme d'une infrastructure standardisée basée sur XML et sur des Services Web utilisés pour supporter des processus métiers*. Ces auteurs proposent une définition très technique de SOA, laquelle est particulièrement attachée à une seule technologie. Cependant, pour apporter plus de clarté, le modèle de référence d'OASIS (*Organization for the Advancement of Structured Information Standards)* définit l'architecture SOA comme *un paradigme pour l'organisation et l'utilisation de capacités de diffusion qui peuvent être sous le contrôle de différents auteurs*. *Il fournit un moyen uniforme pour offrir*, *découvrir*, *interagir avec et utiliser les capacités afin de produire les effets désirés en conformité avec les prés conditions et les attentes mesurables* (MacKenzie et al., 2006). Selon ce modèle, illustré par la Figure 7, un fournisseur de services peut publier une interface bien définie sur un répertoire

qui permet à d'autres parties prenantes de le récupérer et de coupler faiblement ce service offert à leurs propres services. Selon cette définition, les caractéristiques principales d'une architecture SOA sont le *faible couplage*, l'*indépendance par rapport aux aspects technologiques* et l'*extensibilité*. Le faible couplage permet de garantir la réutilisation et l'interopérabilité des services. L'*indépendance aux technologiques* est garantie grâce aux contrats d'utilisation qui sont indépendants de la plateforme technique utilisée par les fournisseurs de services. Enfin, l'*extensibilité* est rendue possible par le fait que de nouveaux services peuvent être découverts et invoqués à l'exécution.

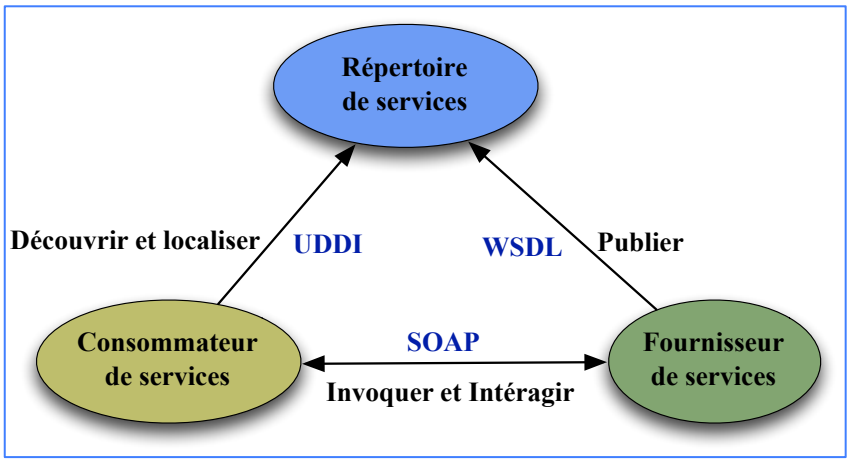


Figure 7. L'Architecture Orientée Service (SOA) et les technologies des Services Web (d'après (O'Sullivan et al., 2002))

L'architecture orientée services se base sur une collection de politiques, principes, interfaces et patrons de conception qui permettent l'intégration des différentes fonctionnalités exposées comme des services (Papazoglou et Heuvel, 2007) (Van Der Aalst et al., 2007). Selon (Papazoglou et Heuvel, 2007), l'architecture SOA fournit la capacité d'adresser les exigences de l'informatique distribuée, à savoir : *indépendance des protocoles*, *faible couplage*, *réutilisation* et *standardisation*. L'architecture SOA est ainsi mise en place afin de garantir, d'une part, la flexibilité dans la maintenance et l'évolution des systèmes, et d'assurer, d'autre part, un niveau élevé d'interopérabilité entres des systèmes hétérogènes et d'adaptation aux changements (Papazoglou et al., 2008).

Cette évolution a encouragé beaucoup d'entreprises à adopter l'architecture SOA afin de permettre une réponse rapide aux changements et une réduction sur le coût de remplacement des systèmes hérités qu'ils avaient, ainsi que l'intégration de nouveaux systèmes basés sur cette architecture permettant à toutes les informations d'être accessibles et partagées par tous les systèmes (Keen et al., 2004) (Minoli, 2008).

L'architecture SOA, comme elle a été définit par (MacKenzie et al., 2006), est utilisée afin de permettre d'une part l'*identification et la publication de services* pour les rendre réutilisables par les autres systèmes et d'autre part la *découverte et la localisation* de services déjà existants. La Figure 7 illustre ainsi les trois principaux acteurs qui interviennent dans cette architecture : le *fournisseur*, le *consommateur* (client) et le *répertoire de services*. Le

fournisseur de services se charge de décrire les fonctionnalités offertes par un service ainsi que les informations nécessaires pour permettre à un client de l'utiliser. Ceci représente la description de services qui sera par la suite publiée par le fournisseur dans un *répertoire de services* (ou annuaire de services). Ce dernier joue le rôle d'intermédiaire entre le fournisseur et le consommateur de services. Ce *consommateur* peut interagir avec le répertoire de services afin de retrouver le service qui répond à ses besoins. Cette étape correspond à la phase de découverte de services, laquelle se base sur les descriptions publiées afin de ne sélectionner que celles qui pouvant satisfaire le client (consommateur de service). Si le client trouve le service désiré, alors il peut mettre en place un contrat avec le fournisseur de services, afin de consommer le service.

Le principe de conception derrière l'architecture SOA est qu'un service est une unité faiblement couplée composée d'une interface et d'une implémentation. L'architecture SOA se caractérise ainsi par la séparation entre la description de service (interface) et sa mise en œuvre (implémentation). L'interface définit l'identité d'un service, ses moyens d'invocation et ses capacités fonctionnelles, tandis que l'implémentation représente la mise en œuvre des opérations internes que le service doit exécuter. Cette séparation a permis aux fournisseurs et aux consommateurs de services d'être faiblement couplés. En outre, les services peuvent être facilement réutilisés. Parce que les interfaces de services sont indépendantes de la plateforme et que la mise en œuvre est transparente pour les consommateurs de services, un client devrait être capable d'utiliser le service à partir de n'importe quel terminal de communication en utilisant n'importe quelle plateforme informatique, système d'exploitation et n'importe quel langage de programmation. Ainsi, l'architecture SOA introduit essentiellement une *nouvelle philosophie pour le développement d'applications distribuées, où les services peuvent être découverts, composés, publié, réutilisés, et invoqués au niveau de l'interface, indépendamment de la technologie spécifique utilisée en interne pour implémenter chaque service* (Granell et al., 2010).

Depuis son apparition, l'*orientation service* n'a cessé de progresser dans la perspective de satisfaire au mieux les besoins des utilisateurs. Cette évolution peut être analysée selon différentes visions, à savoir la vision *technologique*, la vision *sémantique* et la vision *intentionnelle*. Nous les détaillons dans la section suivante.

3.4. L'ORIENTATION SERVICE SOUS SES DIFFERENTES FORMES

L'orientation service, telle que présentée dans les sections précédentes, n'a cessé d'évoluer. Cette notion a été impactée par une vision purement technologique, faisant apparaître les *services Web* afin de garantir l'interopérabilité et l'intégration des fonctionnalités métiers dans le Web. Cette vision a été étendue, par la suite, avec l'appariation du Web Sémantique. Nous avons assisté à l'émergence des *services sémantiques* qui se basent sur une description sémantique, plus expressive et plus compréhensible. Cette vision vient répondre aux lacunes en termes de recherche d'information et d'extraction de l'information pertinente des pages Web, par exemple. Nous avons vu apparaître également une vision intentionnelle qui place la notion de service à un niveau d'abstraction plus élevé, dans lequel

le service est conçu afin de satisfaire une ou des *intention*(s) (but) précis de l'utilisateur. Cette vision soulève l'importance d'exploiter la notion d'intention dans la description des services, représentant l'émergence des *services* dits *intentionnels.* Ces services sont décris selon l'intention qu'ils arrivent à satisfaire, laquelle représente ce qu'attend l'utilisateur de l'exécution du service.

Dans la suite de cette section, nous discuterons ces visions technologiques, sémantiques et intentionnelles de la notion de service.

3.4.1. Services Web : vers une vision technologique

Depuis son apparition, la notion de *services Web* a été présentée comme la technologie la plus prometteuse qui va permettre d'assurer l'*interopérabilité* et l'*intégration* des fonctionnalités métiers dans le Web. Les services Web représentent une suite logique des intergiciels, comme DCE (*Distributed Computing Environment*) (Harold et Lochart, 1994), CORBA (*Common Object Request Broker Architecture*) (Pope, 1998) et DCOM (*Distributed Component Object Model*) (Redmond, 1997), vers plus d'*interopérabilité* (chose que ces intergiciels n'ont pas réussis à garantir). Cette interopérabilité peut être assurée grâce à la plateforme Web et aux différents standards utilisés.

Dans la littérature et à travers le consortium W3C, nous observons qu'il existe diverses définitions de la notion de *services Web* allant des plus génériques, qui présentent un *service Web* comme une application modulaire accessible à d'autres applications sur le Web (Almeida et Menasce, 2002) (Curbera et al., 2001), au plus spécifiques, qui mettent en évidence les technologies et leurs rôles pour mettre en œuvre un service Web (Booth et al., 2004). Par exemple, le consortium OASIS considère un service Web *comme une application métier modulaire, formant un tout qui a des interfaces basées sur des standards, orientées Internet et ouvertes (OASIS, 2001)*. Cette définition présente les services Web en mettant l'accent sur les standards utilisés sur le Web et sur les interfaces qui permettent d'invoquer les services ouvertement.

Par la suite, le consortium W3C (Austin et al., 2002) a proposé une définition plus précise des services Web. Pour la W3C, un service Web représente *une application logicielle identifiée par une URI (Uniform Resource Identification), dont les interfaces et les liaisons sont définies, décrites et découvertes comme des objets XML. Un Service Web supporte les interactions directes avec d'autres agents logiciels en utilisant le passage de messages basé sur XML via les protocoles de l'Internet.* Austin et al. (Austin et al., 2002) mettent ainsi l'accent sur la façon dont les services Web doivent fonctionner. Dans cette définition, un service Web est définit en fonction de son moyen d'*identification* (URI), de son *interopérabilité* à travers les interfaces et les liaisons en XML et des *interactions* possibles (au travers de protocoles basés sur XML). Nous pouvons observer à ce niveau l'importance du standard XML dans cette dernière définition. Toutefois, ceci peut apporter une certaine confusion dans la mesure où n'importe quelle application sur le Web qui soit fondée sur des technologies basées sur XML peut être considérée comme un Service Web.

Une nouvelle définition a ainsi été proposée par le consortium W3C (Booth et al., 2004) mettant plus l'accent sur l'ensemble des technologies utilisées. Ces auteurs définissent un Service Web comme *un système logiciel conçu pour supporter l'interaction interopérable de machine à machine sur un réseau. Il possède une interface décrite dans un format exploitable par la machine, i.e. décrite en WSDL (Web Services Description Language). D'autres systèmes interagissent avec le Service Web d'une façon prescrite par sa description en utilisant des messages SOAP (Simple Object Access Protocol), typiquement en utilisant HTTP (HyperText Transfer Protocol) avec une sérialisation XML en même temps que d'autres normes du Web* (Booth et al., 2004). Cette définition apporte un niveau plus élevé de granularité en précisant les technologies et leurs rôles pour mettre en œuvre un service Web. Elle spécifie *WSDL* (Christensen et al., 2001) comme la technologie utilisée pour la description des Services Web et *SOAP* (Mitra et Lafon, 2007) comme la technologie de communication et de messagerie assurant l'interaction avec le Service Web en question. Toutefois, nous remarquons que cette définition ne précise pas, dans le même contexte, la technologie utilisée pour permettre à un consommateur de services de rechercher un service, et au fournisseur de services de publier son service dans le répertoire de services. Une réponse à cela est déjà présentée par (OASIS, 2001) à travers la technologie UDDI comme une API et un modèle de données standard permettant l'enregistrement des informations sur les entreprises et fournisseurs de services.

Nous pouvons dégager ici, une vision des services totalement concentrée sur les technologies utilisées, à savoir : (i) *WSDL* pour la description des services ; (ii) *UDDI* pour la publication des services dans un répertoire de services et pour la localisation des services recherchés ; et (iii) *SOAP* pour communiquer et transporter les données. Nous résumons ces technologies dans les prochaines sections.

La Figure 7 illustre l'usage de ces technologies de services Web par les différents acteurs de l'architecture SOA :

- Le *fournisseur de services* décrit les services en WSDL. Par la suite, il enregistre et publie ces services à l'aide d'un répertoire UDDI en fournissant certains détails sur le service, tels que les informations de contact sur le fournisseur, le pointeur vers le service et sa description WSDL ;
- Une fois le répertoire est rempli d'informations sur les services, le consommateur de services (client) peut accéder au répertoire ;
- Les requêtes UDDI sont envoyées par le consommateur de services. Si un service pertinent est trouvé, alors le répertoire fournit au consommateur les informations nécessaires pour accéder au service ;
- Le consommateur, à ce stade, a encore besoin de comprendre comment accéder et invoquer réellement le service. Ces informations sont généralement disponibles dans la description WSDL du service. Il peut alors négocier avec le fournisseur le contrat d'accès au service. Ces interactions sont assurées à travers le protocole SOAP.

Le langage WSDL (version 1.0 (Christensen et al., 2001) et version 2.0 (Chinnici et al., 2007)) permet ainsi de *décrire les interfaces des services Web,* en représentant de manière abstraite les opérations que les services peuvent réaliser, et cela indépendamment de l'implémentation qui en a été faite.

L'*Universal Description, Discovery and Integration* (UDDI) (Bellwood, 2002) représente une plateforme destinée à stocker les descriptions des services Web disponibles. Il fournit, d'une part, *au fournisseur de services des mécanismes pour enregistrer leurs services Web*, et d'autre part, *au consommateur de services les mécanismes pour les retrouver*. Le mécanisme de recherche se base sur les mots-clés fournis par les fournisseurs proposant les services.

La plateforme UDDI permet d'héberger un ensemble de descriptions de services Web sur un seul serveur. Toutefois, cette plateforme ne permet pas une recherche efficace de service. Ceci est dû à la simplicité de cette architecture où la sémantique des données est inexistante et où la description des services se limite à de simples *mots-clés* sur lesquels aucun traitement plus approfondi tel que l'approximation n'est possible.

Le *Simple Object Access Protocol* (SOAP) (Mitra et Lafon, 2007) est un format de message basé sur XML pour l'*échange d'informations dans un environnement distribué et décentralisé* (Newcomer, 2002). SOAP est désigné pour gérer le problème de transport et de messagerie dans de larges environnements distribués. Il définit comment organiser les informations en utilisant XML, de sorte qu'il peut être échangé entre les machines.

Cette vision purement technologique ne propose qu'une conception limitée des services et ne permette pas un réel partage des informations et du savoir. Ces technologies permettent de présenter les informations et la connaissance, mais en aucun cas de les rendre facilement utilisables et exploitables. Ainsi émerge un réel besoin d'automatisation de certaines fonctionnalités nécessaires aux Services Web, dont la *découverte, la composition l'invocation et la surveillance de l'exécution des services*.

Ces lacunes ont conduits à la naissance des services Web sémantiques. Cette nouvelle vision apporte avec elle de nouvelles technologies et outils qui peuvent certainement compléter les technologies des services Web. L'objectif ici est, entre autres, de rendre les informations et les services plus compréhensibles afin de faciliter leur réutilisation et exploitation et de répondre au problème de l'automatisation. Nous présentons les services Web sémantique dans la section suivante.

3.4.2. Services Sémantiques : vers une vision sémantique

Dès son appariation, le Web n'a cessé de croître remarquablement. Dans la perspective d'extraire du Web les informations les plus pertinentes et de les réutiliser, il a été proposé le *Web sémantique* (Berners-Lee et al., 2001). Celui-ci se base sur une description formelle de la sémantique des informations et des services. Grâce à cette nouvelle vision du Web, les machines arrivaient à comprendre et potentiellement à satisfaire la demande de l'utilisateur en

se basant sur la sémantique. De manière générale, la notion de services sémantiques vise à créer un Web sémantique qui intègre des services décrit de manière non ambiguë et exploitable par des machines.

Selon le consortium W3C (W3C, 2001), le Web sémantique se rapporte à deux choses différentes. D'abord, il s'agit de *formats communs pour l'intégration des données provenant de sources hétérogènes,* alors que le Web initial s'était focalisé principalement sur l'échange de documents. Ensuite, il s'agit d'un ensemble de *langages pour enregistrer comment les données se rapportent à des objets du monde réel*.

Selon McIlraith et al. (McIlraith et al., 2001), l'association d'une information de nature sémantique au descripteur d'un Service Web a pour objectif la localisation, la composition et l'utilisation automatique des services distribués. La description sémantique fait référence à l'usage des ontologies qui représente une arborescence de concepts liés entre eux avec des relations bien spécifiques. Ainsi, les recherches dans le domaine du Web Sémantique se sont plus accentuées sur la définition de langages et d'outils pour la représentation des informations et des services de telle sorte qu'il soient facilement partagés, réutilisés, combinés et traités sur le Web. Ces recherches ont conduit à de multiples standards tels que *RDF(S)* (Brickley et Guha, 2004) et *OWL* (Patel-Schneider et al., 2004), ainsi qu'à des outils d'édition comme *Protégé* (Noy et al., 2001) et des raisonneurs comme *Racer* (Haarslev et Möller, 2003) et *Jena* (Carroll et al., 2004). De plus, plusieurs initiatives sont apparues dans l'objectif d'annoter et de décrire sémantiquement des services Web. Ces initiatives ont conduit à de multiples approches facilitant les tâches d'automatisation, dont les principales ont été standardisées par le W3C, telles que OWL-S (Martin et al., 2004), WSMO (Lausen et al., 2005) ou encore WSDL-S (Akkiraju et al., 2005) et SAWSDL. (Farrell et Lausen, 2007).

Dans cette section, nous nous concentrons sur les deux standards OWL-S et SAWSDL, utilisés pour enrichir sémantiquement la description des services. Le standard WSMO sera introduit dans la section suivante (*cf.* section 3.4.2.1).

3.4.2.1. OWL-S (*OWL-based Web service ontology*)

Le langage OWL-S (*Semantic Markup for Web Services*) (Martin et al., 2007), initialement connu sous le nom de DAML-S, permet de spécifier et décrire les services Web de façon compréhensible, non ambiguë et facilement interprétable. Ce langage, basé sur le langage d'ontologie OWL (Smith et al., 2004), est représenté aussi comme une ontologie basée sur OWL, définie afin de décrire les services Web sémantiques. Son objectif est de permettre aux utilisateurs et aux agents logiciels de découvrir, d'invoquer, de composer et de contrôler automatiquement les ressources Web offrant des services (Martin et al., 2004). OWL-S définit les capacités des services Web en trois parties, comme l'illustre la Figure 8. Ces trois parties représentant trois ontologies interdépendantes nommées *« Service Profile »*, *« Process Model »* et *« Service Grounding »* (Martin et al., 2007). Chacune décrit un aspect important d'un service Web sémantique.

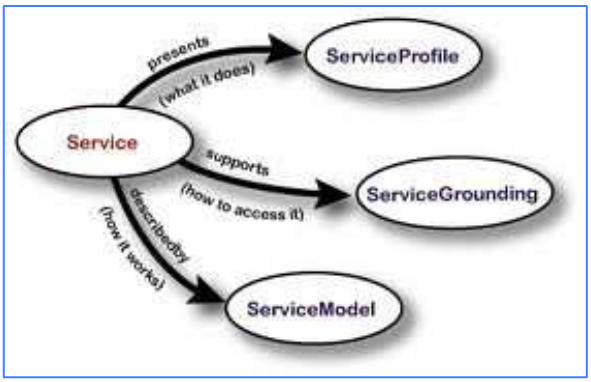


Figure 8. La structure du haut niveau de l'ontologie OWL-S (d'après (Martin et al., 2004))

Le *« service profile »* exprime *ce que le service réalise* (Martin et al., 2004). Il donne une description de haut niveau d'un service, comme l'illustre la Figure 9, à des fins de description, de publication et de découverte de services. Le *« service profile »* est utilisé à la fois par les fournisseurs de services pour publier leurs services et par les consommateurs pour spécifier leurs besoins afin de découvrir le service le plus pertinent.

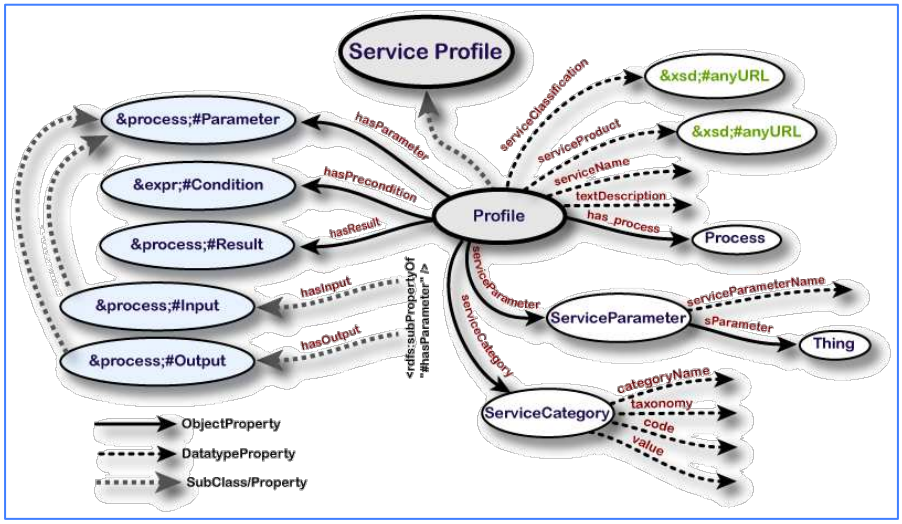


Figure 9. Classe et propriétés de l'ontologie du profil de service de OWL-S (d'après (Martin et al., 2004))

Le *« process model »* répond à la question : *comment est-il utilisé ?* (Martin et al., 2004). Il définit les comportements d'un service en tant que processus défini par les entrées/sorties et décrit comment cela fonctionne. Le « *process model* » est utilisé, entre autres, à des fins de composition de services. En OWL-S, il existe trois types de processus illustrés à la Figure 10 : les processus atomiques (*AtomicProcess)*, les processus simples (*SimpleProcess)* et les processus composites (*CompositeProcess*). Un *processus atomique* représente le niveau le plus fin pour un processus et correspond à l'action qu'un service peut effectuer en une seule interaction. C'est un processus qui peut être directement invocable sans sous-processus. Les *processus composites* représentent des processus qui sont décomposables en d'autres

processus, qui peuvent être eux mêmes des composites. Un processus composite se base sur un ensemble de structures de contrôles telles que : « *sequence* », « *split* », « *choice* », afin de spécifier sa décomposition.

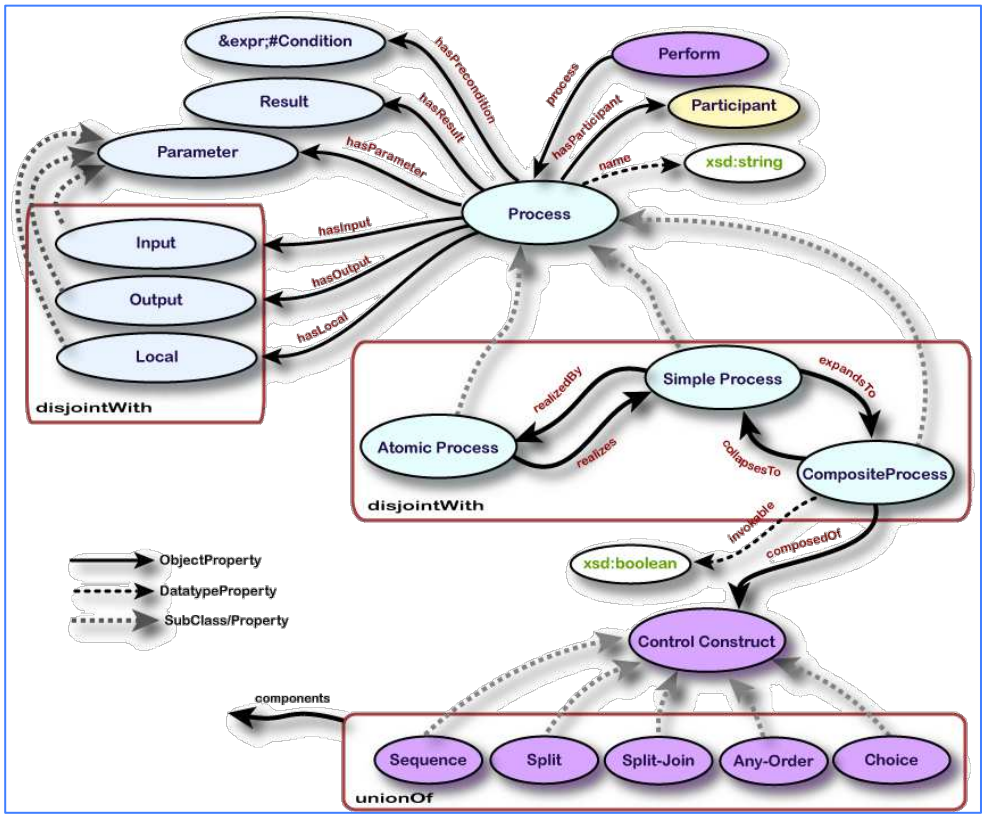


Figure 10. La structure de l'ontologie de haut niveau du modèle de processus de OWL-S (d'après (Martin et al., 2004))

Enfin, le « *service grounding* » expose *comment on peut accéder au service* (Martin et al., 2004). Il indique comment accéder concrètement au service et fournit les détails concernant les formats de messages, les protocoles, entre autres. Cette information est particulièrement utile pour l'invocation automatique de services et elle est souvent exprimée en WSDL.

La description proposée par OWL-S permet une découverte basée sur la sémantique, grâce à des outils et des algorithmes de mise en correspondance, tels que OWLS-Matcher (Jaege, 2001). De plus, OWL-S offre une grande expressivité ainsi que la possibilité d'inférer de nouvelles connaissances grâce notamment aux nombreux moteurs de raisonnements proposés pour OWL, tels que *Jena* (Carroll et al., 2004b), *Pellet* (Parsia et Sirin, 2004) et *Racer* (Haarslev et Möller, 2003). Enfin, même si OWL-S est conçue pour les services Web, elle est riche et suffisamment générale pour décrire tout type de services (Suraci et al., 2007).

3.4.2.2. SAWSDL (Semantic Annotations for WSDL)

L'annotation sémantique de WSDL, nommé *SAWSDL (Semantic Annotations for WSDL)* (Farrell et Lausen, 2007) est une recommandation W3C qui a fait suite à l'approche WSDL-S

(Akkiraju et al., 2005). SAWSDL est une approche utilisant les ontologies pour enrichir la description standard d'un service en se basant sur des mécanismes d'annotation sémantique.

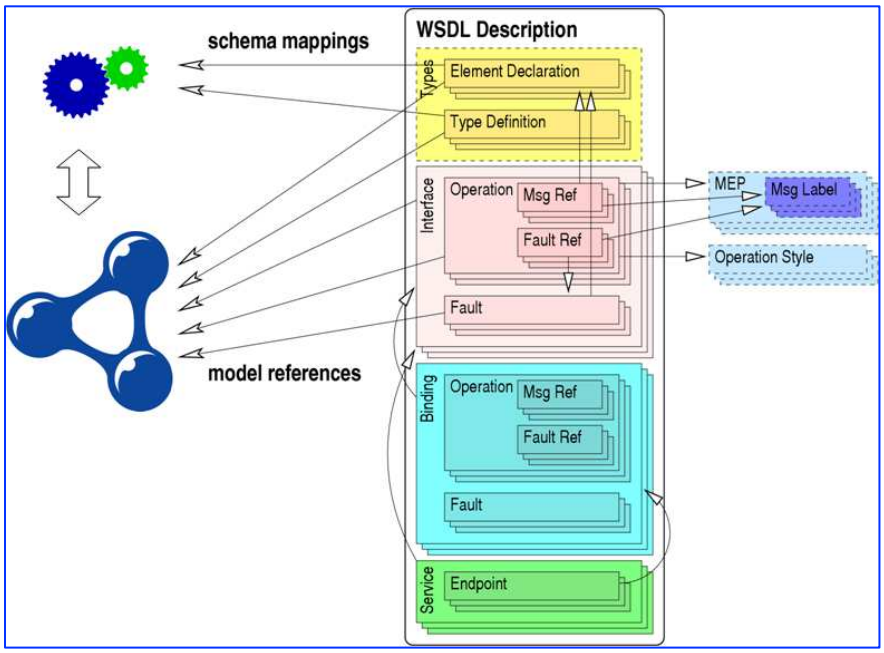


Figure 11. Présentation de SAWSDL avec ces deux formes d'annotations (d'après (Farrell et Lausen, 2007))

A l'encontre de OWL-S, SAWSDL ne définit pas de nouveaux langages pour modéliser sémantiquement les services, mais il fournit un mécanisme de référencement vers des modèles sémantiques souvent externes comme des ontologies.

L'extension SAWSDL, comme l'illustre la Figure 11, se présente sous deux formes d'annotation. La première est *le modèle de référence*. Celui-ci est utilisé pour associer aux interfaces/ports, opérations, entrées, sorties, des concepts sémantiques d'une ou plusieurs ontologies. Il peut être appliqué à n'importe quel élément de WSDL, dans le but d'y associer un ou plusieurs concepts sémantiques. La valeur est un ensemble d'URI (Uniform Resource Identifier), identifiant chacun un fragment de cette description sémantique. En tant que tel, le modèle de référence sert comme un *crochet* où la sémantique peut être fixée (Fensel et al., 2011). Ainsi, il peut être utilisé pour décrire la sémantique des données ou afin de spécifier celle d'une opération d'un service Web. La deuxième forme d'annotation est le *mappage de schémas* lequel spécifie les transformations de données entre les structures de données XML des messages et le modèle sémantique associé. Cette transformation d'une représentation de données à une autre permet d'assurer l'interopérabilité nécessaire entre formats de données hétérogènes lors de l'invocation (*e.g.* contenu d'un message SOAP).

L'inconvénient de l'approche SAWSDL est qu'elle se base sur la simple annotation qui mélange la description technique avec une description de plus haut niveau. Ainsi, plus la taille des informations à annoter devient importante, plus il devient difficile de les analyser.

3.4.3. Services Intentionnels : vers une vision intentionnelle

Les services Web, ainsi que les services Web sémantiques, sont là pour répondre aux requêtes et aux besoins de l'utilisateur. Différentes approches (Lausen et al., 2005) (Kaabi, 2007) ont soulevé ainsi l'importance de l'exploitation de la notion d'intention dans la description des services. Une *intention* représente ce qu'attend l'utilisateur lors de l'exécution d'un service, représentant la vision de l'utilisateur sur les fonctionnalités qu'il désire dans un service (Fensel et al., 2011). Ceci représente une nouvelle vision des services : la *vision intentionnelle*. Cette vision place la notion de service à un niveau d'abstraction plus élevé où le service est conçu afin de conduire à la satisfaction d'une intention de l'utilisateur. Selon (Rolland et al., 2010), cette vision permet de combler le fossé séparant une vision purement technique d'une vision purement métier des services, centrée sur l'utilisateur et ses besoins.

Nous présentons, dans la suite de cette section, la définition de la notion d'*intention*, ainsi que deux approches orientées intentions, à savoir *WSMO* et l'architecture *iSOA* (*Intentional Service Oriented Architecture*).

3.4.3.1. La notion d'intention

Les approches intentionnelles ont été proposées dans le domaine de l'ingénierie des exigences dans la perspective de capturer l'intentionnalité derrière les exigences logicielles (Yu et Mylopoulos, 1998). Dans ce cadre, les intentions représentent une abstraction utile pour décrire les besoins et les attentes des parties prenantes d'un système et elles offrent une façon très intuitive pour obtenir et analyser ces besoins. D'une manière générale, une intention exprime un but, un objectif que l'on souhaite atteindre et que le système doit réaliser. Plihon *et al*. (Plihon et al., 1998) définit une intention comme *quelque chose que certains intervenants espèrent réaliser à l'avenir*.

Selon Antón et al. (Antón et al., 1994), une intention se réfère aux *objectifs de haut niveau du métier, de l'organisation ou du système*. Dans le même contexte, Van Lamsweerde (van Lamsweerde, 2000) rajoute que l'intention se présente comme un objectif que le système doit atteindre à travers une coopération d'agents dans le futur logiciel et dans l'environnement. Cet auteur souligne une distinction entre la notion d'intention et celle d'exigence. Il affirme qu'une *intention* est une *assertion prescriptive que doit satisfaire le système considéré*, alors que l'*exigence* est une *assertion prescriptive que doit satisfaire la partie logicielle du système uniquement* (van Lamsweerde, 2001).

Le terme *intention* a plusieurs significations différentes. Selon (Jackson, 1995), l'intention est une *déclaration « optative », exprimant un état qui devrait être atteint ou maintenu.* En d'autres termes, l'intention représente *l'objectif ou le but que nous voulons atteindre sans dire comment l'exécuter* (Jackson, 1995). Cette intention peut être traduite comme le but qu'un utilisateur souhaite atteindre sans avoir à spécifier comment y parvenir ou encore comme le but à atteindre pour mener à bien un processus, celui-ci étant composé d'une séquence de sous-intentions et de stratégies pour l'atteindre (Kaabi et Souveyet, 2007). Santos et al.

(Santos et al., 2009) définissent une *intention* comme un objectif à atteindre par l'exécution d'un processus présenté comme une séquence d'intentions et des stratégies à l'intention cible.

Même si elles diffèrent, toutes ces définitions nous permettent de considérer l'intention comme *une exigence de l'utilisateur représentant l'intention qu'il souhaite être satisfaite par un service sans dire comment l'exécuter* (Najar et al., 2009). Cette intention représente donc la demande de l'utilisateur quand il est à la recherche d'un service répondant à ses besoins.

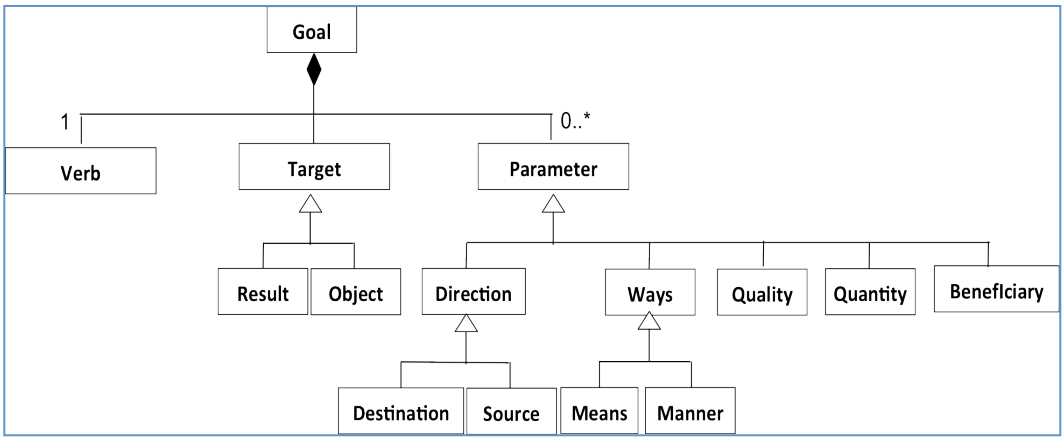


Figure 12. Modèle d'intention (d'après (Prat, 1997))

Avec l'émergence de cette notion d'intention dans le domaine de l'ingénierie des exigences, Prat (Prat, 1997) a proposé une modélisation de cette notion. Ce modèle d'intention est dérivé de l'approche linguistique, laquelle est inspirée par la grammaire des cas de Fillmore (Fillmore, 1968) et des extensions de Dik (Dik, 1989).

Selon ce modèle, illustré par la Figure 12, une intention est décrite sous la forme d'un *verbe* associé à une *cible* et complété par un ensemble de *paramètres* optionnels. Une déclaration d'intention est alors représentée par un verbe, cibles et les différents paramètres qui jouent des rôles spécifiques par rapport au verbe. Dans cette modélisation, le *verbe* expose l'action permettant la réalisation de l'intention, tandis que la *cible* représente soit l'*objet* existant avant la réalisation de l'intention, soit le *résultat* découlant de la satisfaction de l'intention. Les *paramètres* (façon, direction, quantité, qualité et bénéficiaire) sont utiles pour clarifier l'intention et pour exprimer les informations supplémentaires. Le paramètre *direction*, par exemple, caractérise la source et la destination des entités. La *destination* identifie l'emplacement des entités produites par la satisfaction de l'intention, alors que la *source* identifie l'emplacement initial des entités. Le paramètre *façon*, quant à lui, fait référence à l'instrument de la satisfaction de l'intention. Il représente le *moyen* et la *manière*. Le *moyen* indique l'entité qui sert d'instrument pour atteindre l'intention, tandis que la *manière* identifie une approche dans laquelle l'intention peut être satisfaite. Enfin, le paramètre de *qualité* définit une propriété qui doit être atteinte ou maintenue. Par exemple l'intention, I1 = *{#préparer, #proposition, ∅}* est composée du verbe *« préparer »* et de la cible *« proposition »* qui représente le résultat découlant de la satisfaction de l'intention.

La notion d'intention a été utilisée dans différents modèles intentionnels, tels que *KAOS* (Dardenne et al., 1993), *i** (Yu, 1996), *Map* (Rolland et al., 1998) et *Tropos* (Bresciani et al., 2004). Ces modèles offrent une représentation intentionnelle utilisée au début de la phase d'analyse des exigences dans le but d'expliquer le *pourquoi* d'un système logiciel. Elles fournissent des constructions utiles pour analyser les objectifs et les moyens de les satisfaire.

Par exemple, le *framework i** (Yu, 1996) propose une approche orientée agent de l'ingénierie des exigences qui est centrée sur les caractéristiques intentionnelles des agents. Dans ce *framework*, les agents s'attribuent les propriétés intentionnelles et raisonnent sur les relations stratégiques qui les lient. La modélisation orientée intention i* est ainsi utilisée afin de comprendre le domaine du problème.

Tropos (Bresciani et al., 2004), quant à lui, est destiné à soutenir les activités d'analyse et de conception dans le processus de développement logiciel, de l'analyse du domaine d'application jusqu'à la mise en œuvre du système. Dans une perspective intentionnelle, Penserini et al. (Penserini et al., 2007) propose une approche de développement se basant sur la méthodologie de développement *Tropos* (Bresciani et al., 2004). Cette approche cherche à analyser les utilisateurs et à identifier leurs intentions dans la perspective de guider la conception du futur système. Cette conception est composée d'agents logiciels qui ont leurs propres capacités et qui sont destinés à supporter la réalisation des intentions du client.

Le formalisme de la *Carte (Map)* (Rolland et al., 1998) décrit le niveau métier selon une perspective intentionnelle en fournissant les directives nécessaires pour la représentation d'un processus métier. Plus spécifiquement, la carte représente un graphe orienté et étiqueté, dont les *nœuds* représentent les *intentions* et les *liens entre les nœuds* forment les *stratégies*. Une *stratégie* représente une manière de réaliser une intention. Dans ce cadre, l'*intention* est définie comme une condition qui peut être atteinte selon différentes stratégies. Selon (Rolland et al., 2010), la *Carte* permet de capturer la variabilité en mettant l'accent sur la stratégie visant à atteindre une intention et les alternatives possibles pour accomplir la même intention. Cette représentation explicite de la variabilité offerte par les *cartes* est absente dans d'autres formalismes d'ingénierie des exigences tel que *Tropos*.

La notion d'intention a été appliquée, sur différents travaux, à l'orientation de service. Dans la suite de cette section, nous mettons l'accent sur deux intéressantes approches *WSMO* et *ISOA* qui reposent sur la notion d'intention qu'un service est capable de satisfaire.

3.4.3.2. WSMO (Web Service Modeling Ontology)

L'ontologie WSMO (*Web Service Modeling Ontology*) (Lausen et al., 2005) est utilisée afin de migrer la gestion des processus métier (BPM) du niveau IT (technique) au niveau utilisateur (Business) (Super Project, 2006). La notion d'intention est utilisée ici afin de préciser les processus et les tâches pour lesquels les services Web les plus appropriés puissent être découverts dynamiquement.

L'ontologie WSMO se base sur le *framework* WSMF (*Web Service Modeling Framework*) (Fensel et al., 2002), lequel est présenté comme un cadre conceptuel à part entière pour décrire les différents aspects liés aux Services Web. Ce *framework* spécifie les principaux éléments pour décrire les services Web sémantiques. De tels éléments incluent :

- Les *ontologies* qui définissent la terminologie, utilisée lors de la description, en termes de concepts, relations, fonctions, instances et axiomes ;
- La *description d'un service Web* qui décrit formellement les fonctionnalités offertes par le service, en termes de capacité et de méthode d'interaction et en termes d'interface ;
- Les *intentions* qui spécifient ce que l'utilisateur attend du service ;
- Les *médiateurs* qui décrivent les différents éléments permettant de résoudre le problème d'interopérabilité entre les différents composants hétérogènes.

WSMO *(Web Service Modeling Ontology)* (Lausen et al., 2005) représente ainsi une ontologie de modélisation des services Web sémantiques proposant un modèle conceptuel pour ces services. Les concepts de cette ontologie sont décrit formellement par le langage *WSML (Web Service Modeling Langage)* (de Bruijn et al., 2006). Les descriptions WSMO, formalisées en WSML, sont exécutées par l'environnement d'exécution *WSMX (Web Service Modeling eXecution environement)* (Bussler et al., 2005), lequel permet également la découverte, la médiation, l'invocation et l'interopérabilité des services sémantiques.

WSMO est bien connu par son approche basée sur les intentions. Cette approche suppose qu'un utilisateur est à la recherche d'un service afin de satisfaire une intention spécifique. Selon Roman et al. (Roman et al., 2005), une intention décrit les aspects liés aux désirs de l'utilisateur par rapport à la fonctionnalité demandée. Ensuite, Keller et al. (Keller et al., 2004) présentent un mécanisme de découverte de services WSMO reposant sur un processus de mise en correspondance entre l'intention de l'utilisateur et les capacités de ces services. Dans WSMO, l'intention de l'utilisateur et les capacités de services ne sont pas formulées selon un modèle spécifique. Cette information est représentée uniquement comme un ensemble d'*objets*. Par conséquent, WSMO ne permet pas d'identifier le rôle réel que joue chaque objet dans la spécification de l'intention. En effet, les intentions dans WSMO n'ont pas une structure bien définie, tel que le modèle de Prat (Prat, 1997) présenté précédemment. Ceci empêche l'exploration de la sémantique propre à chaque objet formant l'intention.

Enfin, comparé à OWL-S, WSMO ne dispose que de quelques outils d'édition, tel que WSML Editor (Kerrigan, 2005) et pas d'autant de moteurs de raisonnement puissants. Ainsi, le développement d'outils pour WSMO s'avèrent une tâche plus difficile à développer que OWL-S, puisque OWL-S s'appuient sur le langage RDF et OWL qui sont plus utilisés que WSML (M'bareck et Tata, 2008).

3.4.3.3. L'architecture iSOA

Rolland et al. (Rolland et al., 2008) soulèvent le problème de la correspondance entre les besoins exprimés par les utilisateurs, à un niveau élevé, et les services logiciels, exprimés à un bas niveau. Selon ces auteurs, afin d'atteindre le vrai potentiel de SOA au niveau de l'entreprise, il est nécessaire de combler le fossé entre les services métiers de haut niveau, compréhensibles par les acteurs métiers, et les services logiciels de bas niveau, compréhensible par le personnel technique. Ces services métiers doivent être exprimés d'une manière intentionnelle, en termes d'objectifs et de stratégies pour pouvoir les atteindre. Ceci représente une approche intentionnelle de description de services proposée par (Kaabi, 2007) (Rolland et al., 2010), appelée l'*architecture orientée service intentionnel* (iSOA - *intentionnal Service Oriented Architecture*).

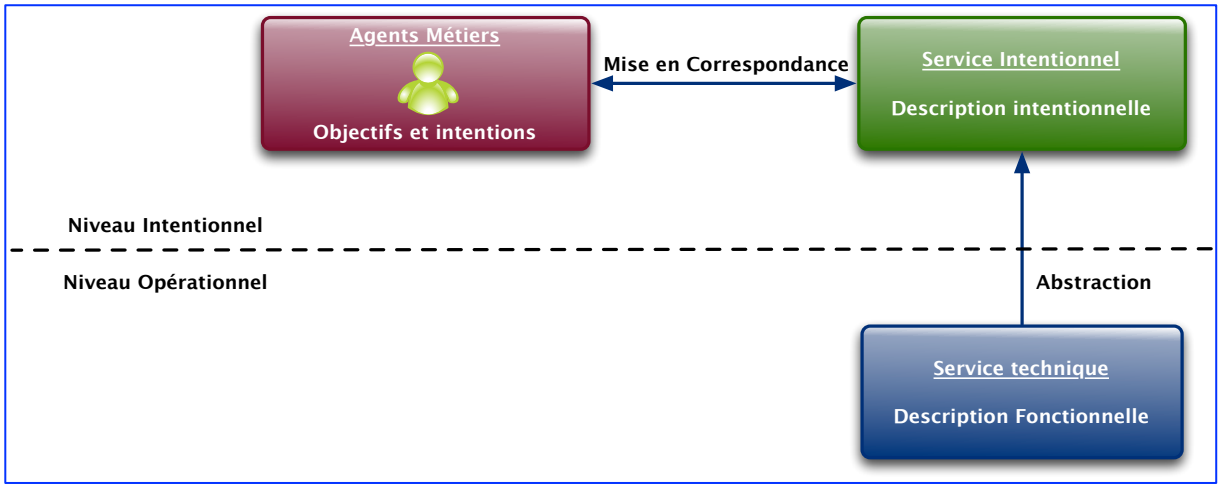


Figure 13. Le service intentionnel : la correspondance entre l'agent métier et le service logiciel (d'après (Kaabi et Souveyet, 2007))

L'architecture iSOA introduit la migration de l'architecture SOA centrée sur les *fonctionnalités* vers une architecture équivalente centrée sur les *intentions*. En réalité, iSOA représente une surcouche intentionnelle de l'architecture SOA, laquelle représente une approche *« top-down »*. Cette approche se base sur deux niveaux d'abstraction, à savoir le *niveau opérationnel,* décrivant les services logiciels, et le *niveau intentionnel,* décrivant les services intentionnels. La Figure 13 illustre cette correspondance entre l'agent métier (le haut niveau) et le service logiciel (le bas niveau). En effet, l'architecture iSOA permet d'abstraire les détails du service technique et de ses fonctionnalités, grâce au concept de *service intentionnel*. Ce service intentionnel correspond aux exigences fonctionnelles des agents métiers : il se concentre sur l'essence du service, à savoir l'intention qu'il permet d'atteindre. Ce service intentionnel exprimé à un haut niveau d'abstraction, se réfère à l'intention qu'il permet de satisfaire plutôt qu'aux fonctionnalités qu'il offre. Ces services, appelés *services intentionnels*, sont exprimés en termes d'intentions et de stratégies pour les atteindre. De plus, iSOA supporte l'opérationnalisation du service intentionnel vers les services logiciels.

Selon Kaabi et Souveyet (Kaabi et Souveyet, 2007), l'architecture iSOA permet une prise en compte directe des objectifs métiers de l'utilisateur. En faisant la distinction avec SOA, dans iSOA les *agents métiers* viennent remplacer les agents logiciels au niveau des interactions, et la *description intentionnelle des services* vient remplacer la description technique. L'*agent métier* qui cherche un ou plusieurs services coïncidant avec ses besoins effectue ainsi une recherche basée sur les intentions métier. En iSOA, le *fournisseur métier* décrit le service de manière intentionnelle et le publie dans un *répertoire de services intentionnels*. La découverte de service est alors guidée par l'intention, en faisant correspondre l'intention de l'utilisateur avec les intentions que les services disponibles peuvent satisfaire. Afin d'assurer la puissante mise en correspondance de l'intention, celle-ci est formulée selon le modèle de Prat (Prat, 1997) présente dans la section 3.4.3.1.

Dans le cadre de l'architecture iSOA, Kaabi et Souveyet (Kaabi et Souveyet, 2007) et Rolland et al. (Rolland et al., 2010) ont proposé un modèle de service intentionnel (ISM – *Intentional Service Model*) qui permet de décrire un service intentionnel selon trois différents aspects : l'*interface*, le *comportement* et la *composition*. L'*interface* expose les principales caractéristiques du service intentionnel aux clients potentiels, permettant sa réutilisation dans un processus de composition de services. Le *comportement* correspond aux pré/post-conditions, alors que la *composition* permet de distinguer les services selon leur granularité (*agrégat* ou *atomique)*.

Selon Rolland et al. (Rolland et al., 2010), le *service atomique* est celui qui ne peut être composé d'autres services intentionnels. Il est lié aux intentions opérationnelles qui peuvent être satisfaites par les services fonctionnels de l'architecture SOA. Ainsi, l'opérationnalisation de service intentionnel atomique peut être assurée par l'exécution d'un service logiciel ou d'une composition de services logiciels. En revanche, les *services agrégats* ont des intentions de haut niveau qui doivent être décomposées jusqu'à ce que des services intentionnels atomiques se trouvent. Cette agrégation est déterminée à partir des liens d'affinement *et*/*ou* de l'intention du service.

La composition intentionnelle admet deux types de services agrégats : un *service composite* et un *service variable.* D'un côté, le *service composite* reflète la relation de précédence ou de succession entre les intentions qu'il satisfait (reliées par un lien *Et*). Dans ses travaux, Kaabi (Kaabi, 2007) propose trois types de services composites. Le premier représente le *service séquentiel,* défini lorsqu'un ordre dans l'appel des services composants est nécessaire. Le deuxième illustre le *service parallèle* qui peut être réalisé selon n'importe quel ordre. Le troisième représente le *service itératif* défini lorsqu'un service intentionnel peut être exécuté récursivement. De l'autre côté, le *service à variation* correspond à différentes manières de réaliser une intention (reliées par un lien *OU*). Kaabi (Kaabi, 2007) propose trois types de services à variation. Le premier représente le service *à choix alternatif* qui exprime un choix parmi plusieurs services composants, lesquels sont mutuellement exclusifs. Le deuxième porte sur le service à *choix multiple* qui exprime un choix non-exclusif entre les services composants. Le dernier illustre le service *multi-chemin* qui introduit une variation portant sur des enchaînements alternatifs d'intentions.

Néanmoins, nous plaidons que cette vision ne tient pas compte de l'évolution de la technologie des services, qui peut se tenir maintenant aux petits logiciels encapsulant des fonctionnalités réutilisables, ainsi que pour de grands systèmes hérités *(Legacy System)*, dont le processus complexe est caché par des technologies telles que les services Web ou ESB (Roshen, 2009). En considérant que seuls les services atomiques peuvent être opérationnalisés à travers le service logiciel, l'architecture iSOA limite la réutilisation de ces systèmes hérités dans une approche intentionnelle. En fait, les systèmes hérités englobent souvent des processus complexes, qui subsument les satisfactions de multiples intentions ou une variabilité intensive sur leur satisfaction. Ces systèmes peuvent être comparés aux intentions agrégats, mais ils ne peuvent pas être assimilées à des intentions atomiques simples.

Les trois visions que nous venons de discuter (technique, sémantique et intentionnelle) représentent le fondement de divers travaux portés sur les systèmes orientés services. Les nouveaux Systèmes d'Information Pervasifs peuvent ainsi s'inspirer de ces trois visions pour adopter une orientation service. Cependant, ceux-ci seront confrontés à différents challenges, que nous discuterons dans les sections suivantes. Nous nous focalisons ensuite sur certaines approches de découverte et de prédiction de services dans les différentes visions.

3.5. Les challenges pour les systemes d'Information Pervasifs Orientes Services

Les Systèmes d'Information Pervasifs (*cf.* section 2.4) se présentent comme une nouvelle génération des SI dans un environnement pervasif hautement dynamique et hétérogène, dans laquelle l'*interopérabilité* est devenue une nécessité à tous les niveaux. Ainsi, la vision technologique de l'orientation service (*cf.* section 3.4.1) permettra de répondre à cette problématique d'interopérabilité. Ces SIP se doivent, de plus, d'être capable d'intégrer, de réutiliser et d'exploiter facilement et d'une manière compréhensible l'ensemble des services qui sont disponibles. Ils doivent fournir des mécanismes de découverte et de composition de services ne nécessitant pas forcément l'intervention d'un utilisateur. Dans ce cadre, la vision sémantique (*cf.* section 3.4.2) pourrait être vue comme une solution pour aider les SIP à répondre à ce besoin. Finalement, les SIP doivent être conçus dans la seule perspective de répondre aux besoins des utilisateurs de la manière la plus appropriée. Ainsi, une prise en compte des besoins des utilisateurs s'avère un principe fondamental de cette nouvelle génération de SI. La vision intentionnelle (*cf.* section 3.4.3) peut contribuer dans ce cadre en mettant en avant les besoins des utilisateurs qui sont formulés sous forme d'intention.

Dans la suite de cette section, nous présentons un ensemble de challenges auxquels les SIP doivent répondre. Nous nous focalisons sur les approches de découverte et de prédiction de services que nous estimons primordiales pour un SIP transparent et proactif.

3.5.1. Les challenges

L'Informatique Pervasive a apporté aux SI de nouveaux défis liés au dynamisme, à l'hétérogénéité et à l'accès ubiquitaire à l'environnement (*cf.* section 2.2.1). Elle a apporté

également de nouveaux challenges dans l'usage des services au sein des SIP. Ces challenges concernent les aspects suivants :

- ***La découverte dynamique des services*** : les SIP doivent satisfaire au mieux les besoins des utilisateurs mobiles. Ceci nécessite la prise en compte de l'utilisateur et de ses besoins ainsi que de son contexte courant, afin de lui proposer le service le plus approprié en toute transparence. La découverte de services est ainsi un des défis majeurs des SIP. Ce mécanisme doit assurer un certain niveau de transparence et d'efficacité tout en s'adaptant au contexte (gestion de l'hétérogénéité et du dynamisme) et en prenant en considération les besoins de l'utilisateur (afin de garantir une meilleure compréhension de l'utilisation réelle des services). Le challenge ici est de retrouver le mécanisme le plus adéquat pour répondre à tous ces besoins ;
- ***La composition dynamique des services*** : face à l'hétérogénéité des services, les SIP se retrouvent devant le challenge de développer des modèles, des techniques et des algorithmes afin de composer dynamiquement les services hétérogènes et de les exécuter d'une manière transparente tout en prenant en considération l'adaptation aux changements fréquent de l'environnement ;
- ***La prédiction (recommandation) dynamique des services*** : un des challenges des SIP est d'introduire des techniques de recommandation dans la perspective d'augmenter le caractère dynamique et proactif du système en proposant à l'utilisateur le service le plus approprié qui pourra l'intéresser. Un utilisateur, même dans le cadre d'un SIP, peut suivre un modèle de comportement qui évoluent avec le temps. Ces schémas de comportement représentent certaines habitudes de l'utilisateur lorsqu'il interagit avec son système. La prise en compte de ces habitudes nous permet d'anticiper ses besoins, et ainsi rendre les SIP plus proactifs. Le défi est de faire ceci de la manière la plus transparente possible. Il faut donc retrouver les moyens pour mieux comprendre les besoins des utilisateurs et observer le contexte dans lequel ils évoluent et sollicitent certains services. L'objectif ici est de trouver le mécanisme qui va permettre d'offrir une meilleure pro-activité au système, notamment par la compréhension de la relation entre la notion de contexte et des besoins des utilisateurs.

Les Systèmes d'Information Pervasifs doivent faire face à l'ensemble des challenges que nous avons évoqués dans cette section. Dans la suite de ce chapitre et dans le cadre de cette thèse, nous allons nous concentrer essentiellement sur les deux challenges : la *découverte* et la *prédiction* de services. Nous nous intéressons ainsi aux différentes approches mises en place par les systèmes orientés services soient-elles techniques, sémantiques ou intentionnelles.

3.5.2. La découverte de services

Durant les dernières années, de grands efforts de recherche ont été menés sur le sujet de la découverte des services. Effectivement, la pertinence d'un mécanisme de découverte de

services dépend de *comment son algorithme de mise en correspondance* (*matching*) *permet d'aller au-delà de ce que fournissent déjà les mécanismes standards comme UPnP, Jini, etc*.

Ce sujet a ainsi été largement traité selon une vision sémantique. Différents travaux, tels que (Paolucci et al., 2002) (Klusch et al., 2006) (Martin et al., 2007), ont concentré leurs efforts sur la mise en correspondance sémantique entre les capacités d'un service et la requête d'un utilisateur. Ces travaux ont servi de base pour d'autres travaux, tels que les travaux dans le domaine de la découverte de services sensibles au contexte (Suraci et al., 2007) (Ben Mokhtar et al., 2008) (Toninelli et al., 2008). Ceux-ci prennent en considération le contexte d'un service et le contexte courant de l'utilisateur lors de la découverte du service le plus approprié. De plus, d'autres travaux suivant une approche intentionnelle ont proposé des mécanismes de découverte de services selon l'intention (Mirbel et Crescenzo, 2010a) (Aljoumaa, 2011) (Olsson et al., 2011). Ceux-ci prennent en considération l'intention qu'un service est capable de satisfaire lors du processus de découverte.

3.5.2.1. La découverte de services sémantiques

Les premiers travaux de découverte de services sémantiques, tels que (Klusch et al., 2006) et (Martin et al., 2007), se sont focalisés sur la mise en correspondance entre les entrées et les sorties pour la découverte du service le plus pertinent face à une requête donnée. Ces auteurs proposent des mécanismes de mise en correspondance *sémantique* (Paolucci et al., 2002) (Martin et al., 2007) et *hybride* (Klusch et al., 2009) en se basant sur les signatures des capacités fournies par les services. Ces mécanismes comprennent notamment l'identification des relations de subsomption entre les concepts décrivant les entrées et les sorties d'un service (Zaremski et Wing, 1995). Ces relations, semblables aux relations d'héritages, permettent de relier des concepts plus spécifiques à des concepts plus généraux explorant ainsi les hiérarchies entre les concepts dans une ontologie.

3.5.2.1.1. L'approche proposée par Paolucci et al.

Une des approches les plus connues de découverte de services sémantiques a été proposée par Paolucci et al.(Paolucci et al., 2002) et Sycara et al. (Sycara et al., 2003). Ces auteurs se basent sur une description de services en DAML-S (OWL-S), dans laquelle un profil de service est décrit afin de refléter les fonctionnalités qu'il souhaite fournir à la communauté. Ils proposent, ensuite, un mécanisme de découverte de services se basant sur un algorithme de mise en correspondance sémantique entre une capacité demandée, décrite sous la forme d'un ensemble d'entrées fournies et de sorties requises, avec un ensemble de capacités fournies, décrites elles aussi sous la forme d'un ensemble d'entrées et de sorties. Cet algorithme se base sur une ontologie DAML (OWL), dans laquelle les entrées et les sorties d'un service sont sémantiquement décrites comme des concepts dans cette ontologie. En se basant sur cette ontologie, le processus de mise en correspondance peut faire des inférences sur les relations hiérarchiques de subsomption, conduisant ainsi à une mise en correspondance sémantique malgré les différences syntaxiques. Dans cet algorithme, Paolucci *et al.* (Paolucci et al., 2002)

analyse les correspondances entre les capacités des services, fournies et demandées, selon quatre niveaux distincts :

- ***Exact*** : si le concept demandé correspond exactement au concept proposé ou s'il représente une sous classe directe du concept proposé ;
- ***Plug-In*** : si le concept proposé subsume celui qui est requis et celui-ci n'est pas une sous-classe directe du premier ;
- ***Subsume*** **:** si le concept demandé subsume le concept proposé et celui-ci n'est pas une sous-classe directe du premier ;
- ***Fail*** : s'il n'y a aucune relation entre les deux concepts.

Dans cet algorithme, Paolucci *et al.* (Paolucci et al., 2002) reposent sur une première étape de mise en correspondance entre les sorties. Cette étape retourne, pour chaque sortie évaluée, un score représentant le degré de mise en correspondance obtenu. Ainsi, le service ayant le score le plus élevé sera sélectionné à la fin de cette étape. Toutefois, selon cet algorithme, la mise en correspondance entre les entrées n'est effectuée qu'en deuxième étape et sous condition que le résultat de la première étape retourne une égalité.

Cette approche a été la base de différentes approches de découverte de services, telles que OWLS-MX (Klusch et al., 2006) que nous présentons dans la section suivante.

3.5.2.1.2. Les approches OWLS-MX, *WSMO-MX et SAWSDL-MX*

Klusch et al. (Klusch et al., 2006) (Klusch et Kaufer, 2008) (Klusch et al., 2009) proposent différentes approches de découverte de services utilisant différents langages de description de services (OWL-S, WSMO et SAWSDL). Ces approches retournent le service le plus pertinent qui répond au mieux à la requête de l'utilisateur. Elles sont dites *hybrides* car elles se basent sur une combinaison de mécanismes de mise en correspondance *syntaxique* et *sémantique*. Chacune de ces approches a abouti à la proposition d'un outil de *Matchmaker : OWLS-MX* (Klusch et al., 2006)*, WSMO-MX* (Klusch et Kaufer, 2008) et *SAWSDL-MX* (Klusch et al., 2009).

Dans les trois approches, l'algorithme de mise en correspondance reçoit comme requête une description du service voulu en OWL-S, en WSMO ou en SAWSDL et retourne en conséquence les services les plus pertinents, qui se rapprochent au mieux de cette description. A chaque service est associé le degré de mise en correspondance et une valeur de similarité syntaxique par rapport à la requête. L'utilisateur peut spécifier le degré de mise en correspondance souhaité, ainsi qu'un seuil pour la valeur de similarité syntaxique. La différence entre les approches réside dans les formats de services utilisés et dans les degrés de mise en correspondance appliqués.

L'approche *OWLS-MX* (Klusch et al., 2006) propose cinq degrés de mise en correspondance en plus des degrés *Exact* et *Fail*, dont trois basés uniquement sur la logique (*Plug-In*, *Subsumes* et *Subsumed-By)* et deux de type hybrides (*Logic-based Fail* et *Nearest-*

neighbor). Le degré *Logic-based Fail* est utilisé lorsque le service ne parvient pas à répondre à la demande en fonction des critères de filtrage basés sur la logique, tandis que le degré *Nearest-neighbor* indique une certaine similarité entre un service *S* et une requête *R* définie comme : $\forall IN_S \exists IN_R : IN_s \geq IN_R$ *ET* $\forall OUT_R \exists OUT_s : OUT_R \geq OUT_S$ *V* $SIM_{IR}(SnR) \geq \alpha$.

Contrairement au précédent, le *Matchmaker WSMO-MX* (Klusch et Kaufer, 2008) utilise WSML à la place de OWL-S. La spécificité de ce *Matchmaker* réside dans l'usage de la notion d'intention *(goal*) dans la mise en correspondance, par l'application récursive de différents filtres sur les pré-conditions et les post-conditions associées à un service. Ces filtres représentent les mises en correspondance *intentionnelles* et *syntaxiques,* ainsi que ceux sur les *relations*, sur les *contraintes* et sur les *paramètres*. Le résultat est une mise en correspondance de l'intention selon sept degrés : *equivalence*, *plug-in*, *inverse plug-in*, *intersection*, *disjonction*, *similarité floue* (mise en correspondance basée sur la non logique), et *neutre* (lorsqu'aucune correspondance n'est déterminée ou lors d'un échec, le seuil de *tolérance* d'un échec de mise en correspondance est déclaré).

L'approche *SAWSDL-MX* (Klusch et al., 2009) accepte, quant à elle, des services spécifiés en SAWSDL. Le processus de mise en correspondance se situe au niveau de l'interface de service en évaluant toutes les combinaisons des opérations d'un service offert et du service demandé. Cette évaluation repose sur le *raisonnement par subsomption* (*exact*, *subsumes* et *subsumedBy*) mais également sur *une mise en correspondance syntaxique* similaire à OWLS-MX. Celle-ci compare la moyenne de similitude entre les vecteurs d'entrées et les vecteurs de sorties pour chaque opération offerte et requise à l'aide des mesures de similarité de texte (*Loss-of-Information, Extended Jaccard, cosinus ou Jensen-Shannon)* (Klusch et al., 2009).

Ces approches précédemment citées (Paolucci et al., 2002) (Klusch et al., 2006) (Klusch et Kaufer, 2008) (Klusch et al., 2009) se caractérisent par une mise en correspondance sémantique entre les services disponibles et la requête de l'utilisateur. Dans la section suivante, nous présentons les approches de découverte de services qui, en plus de leur caractère sémantique, prennent en considération aussi le contexte d'utilisation.

3.5.2.2. La découverte de services sémantiques sensibles au contexte

La sensibilité au contexte est la base pour différentes approches de découverte de services, à l'instar de (Suraci et al., 2007) (Ben Mokhtar et al., 2008) et (Vanrompay et al., 2011). Ces approches se basent, dans leur majorité, sur des descriptions sémantiques des services. La sensibilité au contexte étant une des caractéristiques de SIP (*cf.* section 2.4), ces approches sont ainsi particulièrement pertinentes pour les SIP, puisque ceux-ci doivent adapter leurs offres de services à l'environnement et au contexte d'utilisation.

3.5.2.2.1. L'approche DAIDALOS

Suraci et al. (Suraci et al., 2007) proposent une approche orientée contexte pour la découverte de services Web. Celle-ci s'inscrit dans le cadre du projet européen DAIDALOS

II[1]. Suraci et *al.* (Suraci et al., 2007) considèrent la découverte de services sensibles au contexte comme la capacité d'utiliser l'information contextuelle pour découvrir et sélectionner les services les plus pertinents pour l'utilisateur. Ils considèrent que l'utilisateur et le service ont des exigences sur les informations de contexte dont ils ont besoin pour fonctionner correctement. Un utilisateur peut avoir des exigences sur le contexte d'un service qu'il recherche (disponibilité, localisation, etc.), ainsi que sur le contexte fourni par l'environnement (connexion sans fil, etc.). Un service peut demander à son tour des informations contextuelles sur l'utilisateur (lieu, les capacités du terminal, etc.) et de l'environnement (réseau, etc.).

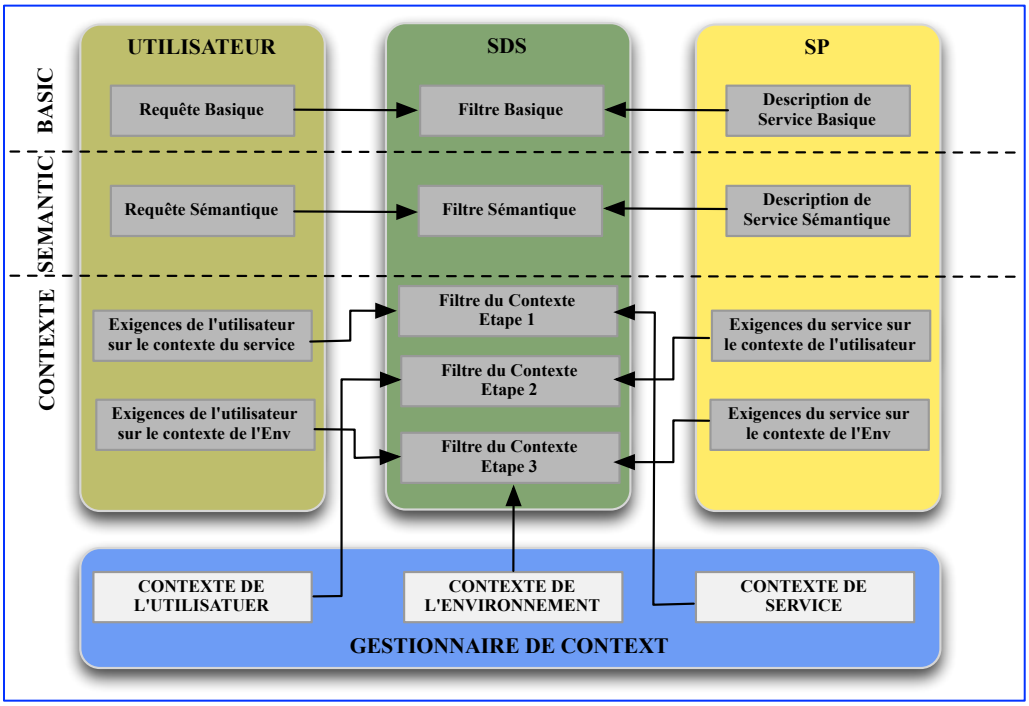


Figure 14. Les trois phases du mécanismes de filtrage lors du processus de découverte de service (d'après (Suraci et al., 2007))

Dans le cadre de cette approche, illustrée par la Figure 14, l'utilisateur débute le processus de découverte de services en fournissant au serveur de découverte de services (SDS – *Service Discovery Server*) sa requête et le pointeur vers sa description de contexte. Cette requête est composée de deux parties : (i) *la requête basique (Basic Query)* exprimée dans un des langages de découverte de services de bas niveau, tels que SLP, UPnP, UDDI, etc. ; (ii) *la requête sémantique (Semantic Query)* exprimée en utilisant un langage de requête sémantique de haut niveau.

Du côté fournisseur, le service est décrit selon le contexte auquel il doit répondre. Celui-ci est décrit et publié par le fournisseur de service dans le gestionnaire de contexte. Ce contexte

[1] *Designing Advanced network Interfaces for the Delivery and Administration of Location independent, Optimised personal Services* : http://www.ist-daidalos.org/

inclut les conditions d'utilisation du service en fonction de la description de l'*utilisateur* (localisation, dispositif, mémoire disponible, taille de l'écran, etc.) et de son *environnement* (la température, le taux de pollution, etc.). Il agit donc comme une contrainte pour l'usage de chaque service. La description d'un service est elle aussi organisée en *description basique,* exprimée en XML ou WSDL, et en *description sémantique,* exprimée en OWL-S.

Après la réception de la requête utilisateur, le SDS invoque le moteur de filtrage de services qui effectue la recherche de services en plusieurs phases, comme l'illustre la Figure 14 :

- Le *filtre de base* compare la requête basique de l'utilisateur aux descriptions basiques des services stockés dans un répertoire de services ;
- Le *filtre sémantique* compare la requête sémantique de l'utilisateur et les descriptions sémantiques des services qui n'ont pas été filtrées lors de la première étape. Le résultat de ce filtre est une liste de services qui répondent aux besoins de l'utilisateur mais sans aucune restriction sur les informations contextuelles ;
- Le *filtre de contexte* compare le contexte associé au service aux attentes de l'utilisateur en termes de contexte de service. Il compare également le contexte de l'utilisateur avec les conditions d'utilisation de chaque service analysé et le contexte de l'environnement avec les exigences de l'utilisateur et du service en termes de contexte de l'environnement.

L'originalité de ce travail réside dans la représentation de contexte associé à un service grâce à l'extension de OWL-S. Plus spécifiquement, le profil du service a été étendu par un attribut *contexte* qui représente une URL pointant vers le contexte réel du service décrit en OWL. Cette séparation entre les deux descriptions facilite la mise à jour des informations contextuelles caractérisées par leur nature dynamique.

3.5.2.2.2. L'approche AIDAS

Toninelli et al. (Toninelli et al., 2008) considèrent que, dans les environnements pervasifs, les utilisateurs ont besoin de services sensibles au contexte adaptés à des paramètres tels que la localisation, l'environnement d'exécution, etc. Ces auteurs proposent alors un mécanisme de découverte de services personnalisé intégrant une représentation sémantique des données de contexte ainsi qu'une mise en correspondances de ces données.

Ce mécanisme de découverte de services a été développé dans le cadre du middleware AIDAS (*Adaptable Intelligent Discovery of Context-Aware Services)*. Celui-ci propose un mécanisme de découverte de services destiné aux utilisateurs mobiles. Le middleware AIDAS exploite la sensibilité au contexte sous forme de *métadonnées*, contenant les informations contextuelles. Ce modèle de métadonnées est composé des *métadonnées de services*, *de l'utilisateur et de son dispositif*. Chacun de ces composants est décrit par un *profil statique* qui contient les informations d'identification, ainsi que les capacités, les exigences et les interfaces des services, et un *profil dynamique* qui décrit les propriétés des services qui

peuvent varier dans le temps. Ce profil dynamique comprend essentiellement des informations sur les conditions d'exploitation du service (état du service).

Dans cette approche, le mécanisme de découverte de services considère en entrée les capacités offertes par les services et celles requises par l'utilisateur et offre en sortie le degré de la similarité sémantique entre eux. Pour chaque capacité requise, l'algorithme est capable de reconnaître les relations de subsomption possibles avec la capacité offerte, à l'instar de Paolucci et al. (Paolucci et al., 2002), à savoir la capacité offerte peut être une instance de la classe de la capacité requise (*exact*), ou une instance d'une classe qui subsume celle-ci (*subsume*) ou une instance d'une classe qui est subsumé par celui-ci (*plug-in*). Ces relations sémantiques sont déterminées en considérant les valeurs de propriétés et les classes des capacités offertes et requises. En cas de correspondance *exacte* pour toutes les capacités du service, alors le service offert est compatible avec la demande de l'utilisateur. Dans le cas où la correspondance n'est pas exacte, la compatibilité est évaluée en fonction des préférences de l'utilisateur. S'il existe une préférence indiquant que la contrainte sur cette propriété peut être assouplie, un *plug-in* ou un *subsume* peut être considéré comme compatible

3.5.2.2.3. L'approche EASY

Similairement aux approches précédentes, Ben Mokhtar et al. (Ben Mokhtar et al., 2008) proposent un middleware sémantique et orienté services, nommé *EASY,* illustré à la Figure 15, pour la découverte et la composition de services dans un environnement pervasif.

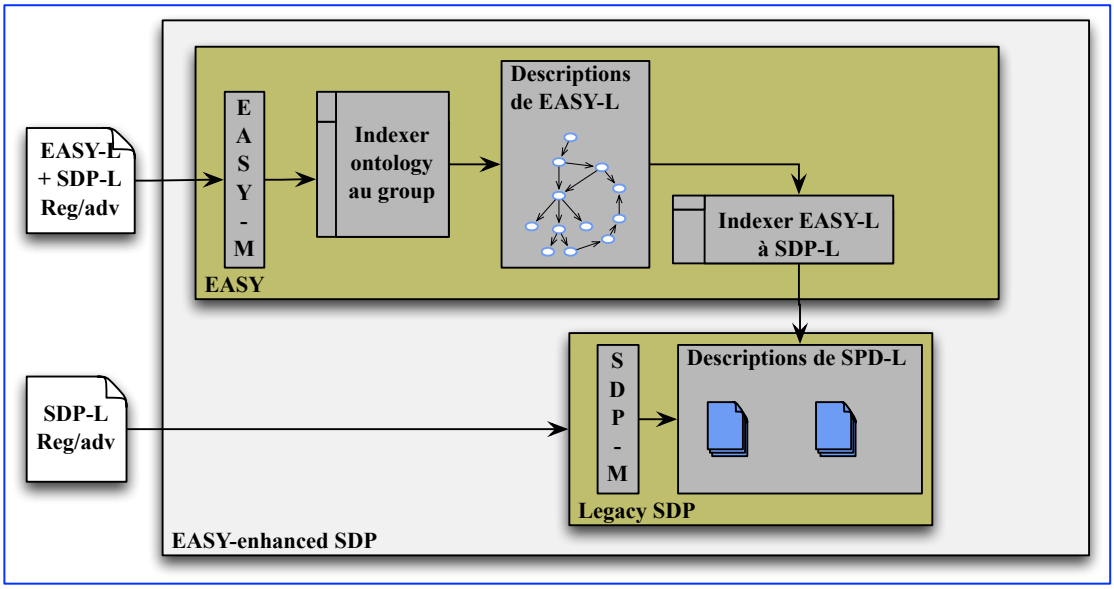


Figure 15. Aperçu de l'architecture du middleware EASY (d'après (Ben Mokhtar et al., 2008))

Cette proposition inclut le langage *EASY-L* basé sur OWL, pour la spécification sémantique, non ambiguë, de propriétés fonctionnelles et non-fonctionnelles des services. De plus, elle présente EASY-M (*EASY-Matching*) qui représente un ensemble de relations de conformité pour la mise en correspondance entre les services en termes de leurs propriétés

fonctionnelles et non fonctionnelles. EASY-M fournit les moyens pour découvrir le service qui correspond au mieux aux exigences non-fonctionnelles des utilisateurs en prenant en considération leurs préférences. Ben Mokhtar et al. (Ben Mokhtar et al., 2008) adoptent des niveaux de mise correspondance similaires à ceux proposés par Paolucci et al. (Paolucci et al., 2002) (*cf.* section 3.5.2.1.1). Ils présentent ainsi leurs trois relations pour la mise en correspondance des propriétés fonctionnelles des capacités requises et des capacités offertes : *ExactCapabilityMatch*, *InclusiveCapabilityMatch* et *WeakCapabilityMach.* La relation *ExactCapabilityMatch* permet de trouver les capacités offertes qui correspondent exactement à une capacité requise. La relation *InclusiveCapabilityMatch* permet de trouver des capacités qui peuvent être plus génériques que la capacité requise, tandis que la relation *WeakCapabilityMach* représente la relation la moins restrictive, où les concepts des capacités requises soit subsument, soit sont subsumés par les concepts offerts. Ceci permet de découvrir des services fournissant des sorties plus spécifiques que celle requises.

En se basant sur EASY-L et EASY-M, le middleware EASY utilise un algorithme d'encodage hors ligne lequel réalise un encodage des concepts des ontologies utilisés et permet ainsi de réduire le coût du raisonnement sémantique sur les ontologies à une comparaison numérique de codes. Selon Ben Mokhtar et al. (Ben Mokhtar et al., 2008), cet algorithme s'appuie sur les nombres premiers et prend en charge un encodage progressif et sans conflit, permettant ainsi de faciliter la réutilisation et l'extension des ontologies existantes. Le middleware EASY se base sur cette technique d'encodage pour d'organiser efficacement les spécifications des services sémantiques dans les répertoires de services. Cette organisation permet de réduire considérablement le nombre de mises en correspondance sémantiques effectuées lors de la découverte de services.

3.5.2.2.4. L'approche proposée par Vanrompay et al.

Vanrompay et al. (Vanrompay et al., 2011) proposent un mécanisme de découverte de services basé sur une mise en correspondance contextuelle, lequel prend en compte l'incertitude des informations de contexte lors du classement des variantes de services. Ce mécanisme se base sur des descriptions de services en OWL-S enrichies avec des propriétés contextuelles. Ces propriétés représentent le *contexte requis du service*, lesquelles sont non fonctionnelles et liées à l'environnement d'exécution le plus adapté pour le service. A l'instar de Suraci et al. (Suraci et al., 2007), la description de contexte requis est incluse dans un fichier XML extérieur référencé à l'intérieur du *profil de service OWL-S*. L'originalité de cette approche réside dans l'analyse de cette description contextuelle sous forme de *graphe*. Dans ce graphe, les objets représentent les concepts et les propriétés, alors que les arêtes représentent les relations reliant ces concepts.

Cette approche par graphe permet de comparer les informations contextuelles en se basant sur des mesures de similarité. Le mécanisme de découverte de services utilise des mesures de similarité locale combinées à des mesures de similarités globales dans le but de comparer les exigences liées au contexte de chaque variante de service avec les valeurs du contexte courant d'exécution. Les *mesures locales* comparent deux nœuds individuellement, en considérant

seulement le concept qu'il représente et ses propriétés. Les *mesures globales* prennent en compte le graphe dans son ensemble, évaluant, par exemple, la proportion des éléments similaires dans les deux graphes. Ces mesures sont basées sur l'analyse des valeurs *moyennes* et des *degrés d'incertitude*. Le degré d'incertitude est représenté comme une métadonnée associée au contexte observé. Plus le degré d'incertitude est faible, plus l'information contextuelle est fiable. Les résultats de ces mesures globales sont utilisés pour classer les services selon leur pertinence dans le contexte courant.

3.5.2.2.5. L'approche proposée par Petit

Petit (Petit, 2010) propose une approche spatiale pour la modélisation et la conception d'un Système d'Information mobile et distribué par l'analyse du contexte. Cet auteur propose un modèle décrivant l'espace géographique d'un système utilisé pour différencier un ensemble de contextes d'exécution. Son but principal est d'aider et de guider les concepteurs à caractériser les évolutions possibles de la mobilité du systèmes lors de sa future exécution. La description de contexte permet de mettre en correspondance les attentes des utilisateurs avec les capacités techniques de la plateforme afin de leur offrir les fonctionnalités les plus adéquates. Dans ce modèle, les composants de la plateforme génèrent une seule région d'exécution définie selon une certaine couverture spatiale. Cette *région d'exécution* présente l'espace dans lequel ce composant agit au sein du système. Ce modèle définit également des *régions d'intérêt* qui sont définies par rapport à des informations relatives à des lieux ou des évènements de l'environnement du système.

Dans ces travaux, Petit (Petit, 2010) propose un *cadre de conception unifié* permettant d'identifier et de prendre en compte les besoins des utilisateurs et leurs attentes futures. Ce cadre se définit selon les deux étapes suivantes : (1) définition des objectifs des utilisateurs ; et (2) identification des étapes nécessaires à l'accomplissement de ces objectifs (tâches). Ce cadre de conception est utilisé comme support pour l'intégration des contextes d'exécution décrits par l'étude géographique de l'environnement et des relations entre régions. Le cadre de conception résultant est dit *cadre de conception étendu*. Il considère les variations du contexte d'exécution au niveau de chaque composant. Il ajoute ainsi une description géographique de l'environnement comme entrée du processus de modélisation. La modélisation des contextes d'exécution est fournie en entrée d'un processus de conception étendu, centré sur l'analyse des tâches des utilisateurs pour l'écriture de scénario.

3.5.2.3. La découverte de services sémantiques intentionnels

En plus des approches sémantiques et sensibles au contexte citées précédemment, une autre vision est proposée par des travaux tels que (Rolland et al., 2010) (Aljoumaa et al., 2011) (Olsson et al., 2011). Ces approches, dites guidées par l'intention, prônent l'importance des besoins de l'utilisateur lors du choix des services.

3.5.2.3.1. L'approche SATIS

L'approche SATIS (*Semantically AnnotaTed Intentions for Services*) (Mirbel et Crescenzo, 2010b) a été proposée dans la perspective d'offrir à des utilisateurs finaux le moyen de représenter leurs démarches de recherche de Services Web pour opérationnaliser des parties d'un processus métier. Cette approche permet, d'une part, aux utilisateurs finaux d'exprimer leurs besoins selon une perspective intentionnelle, et d'autre part, de les aider à rechercher les Services Web disponibles qui correspondent à leurs besoins. Cette approche est dépendante d'un domaine particulier pour lequel des connaissances du domaine et des descriptions de services Web sont disponibles.

L'approche SATIS (Mirbel et Crescenzo, 2010b) repose sur les technologies du Web sémantique pour représenter : (i) les *besoins intentionnels de haut niveau des utilisateurs* ; (ii) *les patrons de spécifications de services Web* ; ainsi que (iii) les *spécifications des services Web*. Dans leur approche, ces auteurs adoptent le *modèle de la Carte* (Rolland, 2007) pour la *description des besoins intentionnels des utilisateurs finaux*. Le modèle de carte met en avant les intentions et les stratégies possibles pour atteindre celles-ci. Celles-ci sont rassemblées dans une ontologie spécifiée en RDFS (Brickley et Guha, 2004), dédiée à la représentation des processus intentionnels et à l'annotation du processus de recherche (Corby et al., 2009). En plus de cette ontologie, l'approche SATIS repose sur deux autres ontologies, une *ontologie OWL-S* utilisée pour la *description de Services Web,* et une *ontologie de domaine*, notamment une ontologie décrivant les images médicales et les traitements d'image associés (Mirbel et Crescenzo, 2010b). Finalement, les patrons de spécifications de service Web sont définis à l'aide du langage de requête SPARQL. Ces patrons de spécifications des services Web sont ainsi modélisés comme des patrons de graphes qui sont projetés sur les graphes des annotations de Services Web (Mirbel et Crescenzo, 2010a).

L'approche SATIS propose ainsi un modèle de réutilisation et de partage de requêtes dans une communauté permettant de construire les démarches d'une façon dynamique. Ces requêtes sont organisées sous la forme d'un ensemble de fragments de démarche de recherche. Un fragment de démarche représente un morceau autonome et cohérent du processus de recherche de Services Web (Mirbel et Crescenzo, 2010a), modélisé sous forme de requêtes *SPARQL*. Chaque fragment permet de supporter l'opérationnalisation d'une partie du processus métier (*e.g.* une chaîne de traitement d'images) à l'aide de Services Web (Mirbel et Crescenzo, 2010b). Ces fragments seront par la suite réutilisés et partagés à l'intérieur d'une communauté d'utilisateurs partageant les mêmes centres d'intérêts dans un domaine.

L'approche SATIS (Mirbel et Crescenzo, 2009) passe par quatre phases, comme l'illustre la Figure 16. La première phase est la ***phase d'élicitation*** dans laquelle les utilisateurs finaux définissent leurs fragments de démarches selon le modèle de la Carte (Rolland, 2007). La deuxième phase représente la ***phase de formalisation*** comportant deux activités. Tout d'abord, elle se charge de raffiner certaines sections (composées d'une intention source, d'une intention cible et d'une stratégie) de la Carte décrite lors de la phase d'élicitation. Ce raffinement permet de détailler la manière d'atteindre une intention cible. Ensuite, elle se

charge de générer les requêtes SPARQL afin de concrétiser chaque section par un service Web approprié ou par un ensemble de spécification de services Web. La troisième phase est la ***phase de fragmentation***, laquelle transforme toutes les spécifications capturées pendant la phase de formalisation en un ensemble de règles (Corby et al., 2009). Finalement, la quatrième phase est la ***phase de population***. Cette phase consiste à dériver les spécifications sémantiques des services Web pour opérationnaliser l'ensemble des intentions et des stratégies associées à la requête en cours de réalisation.

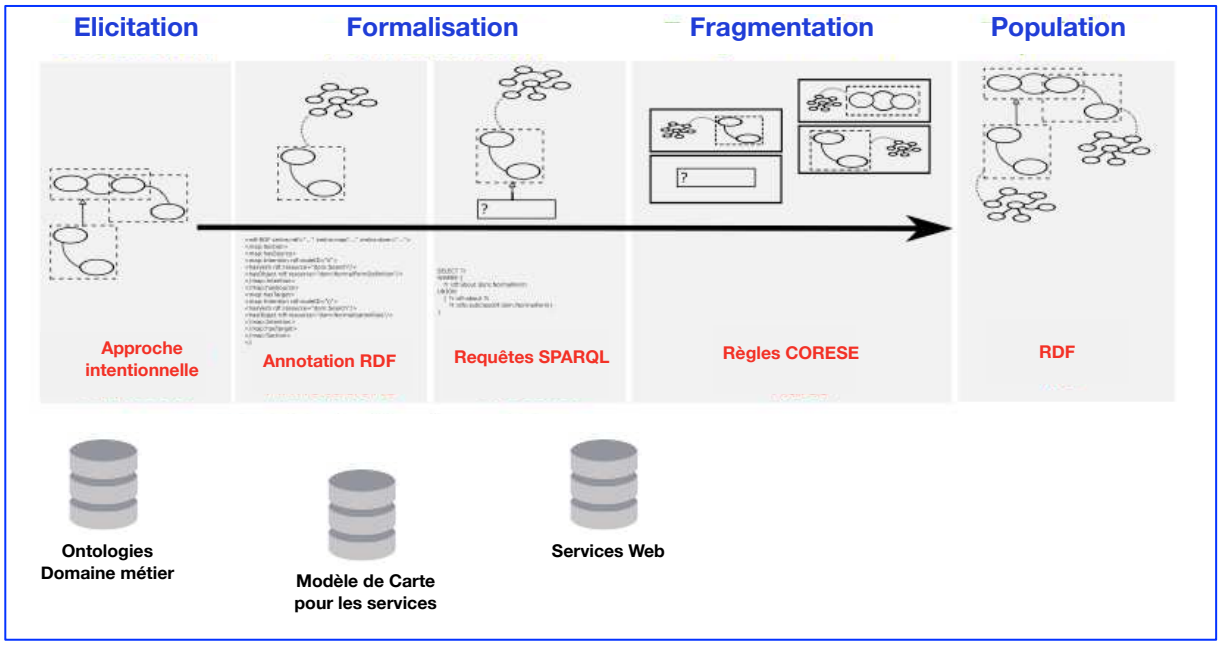


Figure 16. L'approche SATIS (d'après (Mirbel et Crescenzo, 2009))

Dans approche (Mirbel et Crescenzo, 2010b), la phase de recherche comporte deux alternatives : (i) rechercher parmi les besoins intentionnels décrits explicitement dans la mémoire sémantique de la communauté ; ou (ii) créer lui-même une nouvelle spécification de besoin. La création d'une nouvelle spécification consiste à spécifier des besoins intentionnels selon le modèle de la Carte (Rolland, 2007), sous forme d'intentions et de stratégies pour les atteindre. Cette modélisation permet de spécifier des intentions suffisamment précises pour qu'elles soient associées à des patrons de spécifications de Services Web. Par la suite, l'étape de mise en œuvre de la démarche de découverte de Services Web va permettre l'opérationnalisation du processus métier. Quant au mécanisme de recherche de services, il s'appuie sur un moteur de chaînage arrière qui exploite les règles SPARQL implémentant les fragments de démarches. Plus spécifiquement, l'approche SATIS s'appuie sur le moteur sémantique CORESE (Corby et al., 2009), lequel représente un moteur de recherche sémantique basé sur le modèle des graphes conceptuels. CORESE intègre un moteur de *chaînage arrière* exploitant des règles implémentées par des requêtes SPARQL représentant les fragments de la forme *construct-where*. Selon cette formalisation, la clause *construct* est un patron de graphe permettant de construire la représentation RDF de la section de la carte et la clause *where* est un patron de graphe qui représente soit une carte (règle abstraite) soit un critère de recherche de ressources pertinentes (règle concrète).

Le point fort de SATIS est qu'elle s'appuie sur les modèles et langages du Web sémantique pour enrichir la description des besoins des utilisateurs et ainsi proposer des moyens de raisonnement et d'explications des Services Web trouvés pour implémenter un besoin métier.

3.5.2.3.2. L'approche PASiS

Aljoumaa et al. (Aljoumaa et al., 2011) proposent une approche sémantique orientée services basée sur l'architecture iSOA (Kaabi et Souveyet, 2007) (Rolland et al., 2010) (*cf.* section 3.4.3.3). A partir du modèle de services intentionnels *(ISM - Intentional Services Model)*, Aljoumaa et al. (Aljoumaa et al., 2011) proposent l'approche PASiS (*Publishing And Searching intentional Services*) dont l'objectif est de permettre aux utilisateurs finaux d'exprimer leurs besoins sous forme de requêtes basées sur le modèle d'intention de Prat (Prat, 1997) (*cf.* section 3.4.3.1). A l'aide d'un processus de reformulation, l'approche PASiS vise à assister les utilisateurs lors de la formulation de leur requête. Ces auteurs se basent essentiellement sur le principe de guidage méthodologique pour formuler les besoins des utilisateurs sous forme d'intention et découvrir et sélectionner les services intentionnels les plus proches des besoins de l'utilisateur. Cette approche vise ainsi à rendre opérationnels les travaux de (Kaabi et Souveyet, 2007) (Rolland et al., 2010) en implémentant, entre autres, le mécanisme de découverte de services intentionnels de l'architecture iSOA. Ce mécanisme repose sur une description des services en SAWSDL étendue afin de prendre en considération l'aspect intentionnel (Aljoumaa et al., 2011).

Le mécanisme de découverte de services guidé par l'intention, se base sur un ensemble d'ontologies : (i) *iOnto* décrivant les services intentionnels ; (ii) *vOnto* représentant les verbes du domaines ; et (iii) *pOnto* représentant l'ontologie de domaine. Ce mécanisme de découverte applique un algorithme de *mise en correspondance sémantique* entre une requête formulée sous forme d'intention et les intentions des services disponibles dans le répertoire étendu de services. Cette mise en correspondance s'inspire particulièrement de Paolucci et al. (Paolucci et al., 2002). Elle détermine un possible lien d'héritage entre deux concepts dans l'ontologie et ensuite, en fonction du nombre de niveaux hiérarchiques qui les séparent, elle détermine si les deux intentions correspondent.

L'architecture iSOA (*cf.* section 3.4.3.3), comme pour PASiS (Aljoumaa, 2011), considère qu'un service intentionnel est attaché directement à un service technique (une relation 1 à 1). Ces auteurs ne prennent pas en charge non plus la dynamique de l'environnement technique par rapport à la réalisation de l'intention, ni le fait que dans des contextes différents, une intention peut être réalisée par différents services techniques.

3.5.2.3.3. L'approche proposée par Olsson et al.

Olsson et al. (Olsson et al., 2011) défendent également l'utilisation de l'*intention* pour décrire les services selon un nombre arbitraire de niveaux d'abstraction. Ces auteurs proposent une approche de découverte de services orientée intention. Dans cette approche, un utilisateur

doit être capable de spécifier une intention de haut niveau, exprimée en WSML, en termes de *QoS*, d'*indicateurs clés de performance* et, bien sûr, de *fonctionnalités spécifiques*. Cette intention décrit un état souhaité du système, ce qui va permettre au mécanisme de découverte de découvrir les services les plus pertinents qui peuvent satisfaire l'intention de l'utilisateur.

Cette approche se base sur une méthodologie ascendante *(bottom-up)* pour la modélisation sémantique des services Web, en se basant sur l'annotation des fichiers WSDL avec des informations sémantiques pour chaque opération. Ainsi, ces auteurs représentent un service Web comme un ensemble d'opérations annotées sémantiquement. Chaque opération est modélisée comme un ensemble de variables d'entrée et de sortie, et d'états de transition constitués d'une assomption (*assumption*) et d'un effet.

Dans le mécanisme de découverte de services, une intention est mise en correspondance avec les effets des opérations, et les assomptions peuvent alors, à leur tour, être considérées comme des intentions supplémentaires qui infèrent des dépendances entre les opérations. Ce mécanisme se base sur un algorithme de mise en correspondance, également inspiré des travaux de Paolucci et al. (Paolucci et al., 2002). Cette mise en correspondance entre une intention et l'effet d'une opération est déterminée ainsi selon les mêmes niveaux que Paolucci et al. (Paolucci et al., 2002) (*exact*, *plug-in*, *subsume* et *fail*), alors que la mise en correspondance des opérations s'effectue par une vérification de la requête dans le *framework* WSML2Reasoner (Grimm et al., 2007).

De plus, cette approche de découverte de services intègre un algorithme de raffinement d'intention. Cet algorithme décompose une intention en sous intentions plus spécifiques avec lesquelles on peut avoir un résultat de mise en correspondance plus précis. Cet algorithme de raffinement d'intentions permet de déterminer également l'ensemble d'opérations qui sont nécessaires pour l'accomplissement d'une intention. Selon ces auteurs (Olsson et al., 2011), cet algorithme de raffinement permet non seulement la description des services, mais aussi l'amélioration de la performance de la découverte de ces services.

Cette approche se concentre uniquement sur les aspects fonctionnels, permettant de découvrir avec les opérations des services qui devraient être utilisées lors d'une composition.

3.5.2.4. La découverte de services sémantiques sensibles au contexte et intentionnels

Aucun de travaux précédemment cités (Mirbel et Crescenzo, 2010b) (Aljoumaa et al., 2011) (Olsson et al., 2011) ne combine la notion de contexte à celle d'intention, contrairement à Santos et al. (Santos et al., 2009), Ramadour et Fakhri (Ramadour et Fakhri, 2011) et Ma et al. (Ma et al., 2011) qui ont relevé l'importance d'exploiter l'étroite relation entre ces deux notions dans les processus de découverte de services.

3.5.2.4.1. L'approche proposée par Santos et al.

Santos et al. (Santos et al., 2009) proposent *le framework* GSF (*Goal-Based Service Framework*), qui permet une découverte et une composition dynamique de services guidée par l'intention et le contexte. Cette approche se base sur la notion d'intention afin d'analyser les besoins exprimés par les utilisateurs. Pour cela, ces auteurs identifient à priori un ensemble d'intentions spécifiques à un domaine et les différentes tâches qui permettent leur accomplissement. Les services sont aussi associés à ces tâches, permettant une découverte guidée par l'intention. Le concept d'intention est utilisé afin d'exprimer l'objectif de l'utilisateur vis à vis du service.

Le *framework* GSF illustré à la Figure 17 propose ainsi une ontologie des services basée sur les intentions (*GSO*) *(Goal Service Based-Ontology)* décrivant les concepts indépendants du domaine tels que le *service*, le *client*, le *fournisseur*, l'*intention*, la *tâche* et leurs relations. Ces concepts sont par la suite utilisés et spécialisés dans les ontologies de tâche et de domaine. L*'ontologie de domaine* inclut les concepts spécifiques à un domaine, les relations entre ces concepts et les *intentions* valides que les utilisateurs de ce domaine peuvent avoir, tandis que l'*ontologie de tâche* utilise les concepts définis dans l'ontologie de domaine et fournit les définitions spécifiques à ce domaine des tâches et comment elles peuvent être liées à l'accomplissement des intentions de l'utilisateur.

Le *framework* GSF fournit également *une plateforme des services sensibles au contexte* supportant l'interaction entre les fournisseurs et les clients de services. Du point de vue fournisseur, la plateforme prend en charge la publication des descriptions de services. Du point de vue client, celle-ci fournit des mécanismes de découverte, de composition et d'invocation de services. Cette plateforme comprend des composants de gestion de contexte qui se chargent de fournir des informations contextuelles de l'utilisateur. Ces informations représentent des données d'entrée pour les services découverts, et sont utilisées autant pour la sélection de la tâche qui répond à une intention donnée, que pour réduire l'interaction de l'utilisateur avec la plateforme soutenant ainsi un comportement plus autonome.

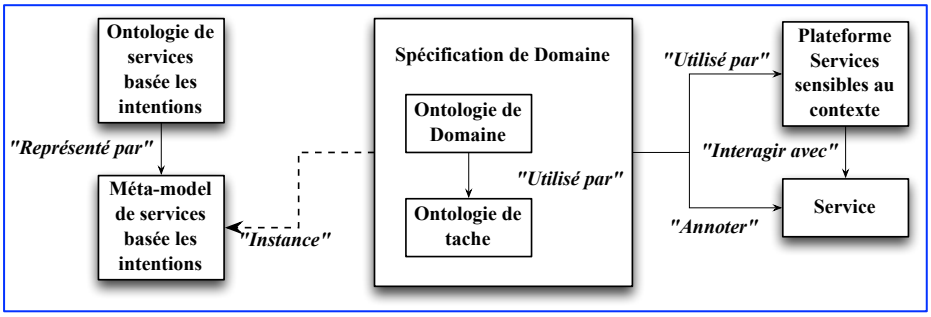


Figure 17. Les principaux composants de GSF (d'après(Santos et al., 2009))

A travers cette plateforme (Santos et al., 2008) (Santos et al., 2009), un client peut soumettre soit une intention, en demandant à la plateforme de lui retourner le service qui satisfait celle-ci, soit directement une tâche. Si l'utilisateur décrit sa demande sous forme

d'intention, alors un processus de découverte de services en deux phases démarre. La première phase effectue une mise en correspondance entre l'intention de l'utilisateur et les intentions définies dans l'ontologie de domaine. Si une intention est trouvée, le processus déclenche la deuxième phase qui va chercher les tâches définies dans l'ontologie de tâche pouvant répondre à cette intention. Ce processus identifie alors les services qui implémentent la tâche satisfaisant l'intention demandée, grâce à une correspondance entre les services et les tâches fixées à l'étape précédente. Par contre, si l'utilisateur choisit de soumettre une tâche, alors la plateforme tente de trouver la mise en correspondance entre la tâche demandée et celles dans l'ontologie de tâches, puis les services qui implémentent les tâches choisies.

Cette approche semble particulièrement restrictive car elle demande l'association au préalable des intentions aux tâches dans l'ontologie de tâches. Par ailleurs, la notion de contexte n'est utilisée que comme un paramètre d'entrée pour les services recherchés, sans avoir une réelle relation avec l'intention. Ces auteurs ne considèrent le contexte que comme un filtre pour la découverte de services, le contexte étant décrit comme une partie des entrées nécessaires aux services, et les intentions comme de simples étiquettes permettant de relier les demandes des utilisateurs aux services.

3.5.2.4.2. L'approche proposée par Ramadour et Fakhri

Ramadour et Fakhri (Ramadour et Fakhri, 2011) proposent une méthode de découverte et de composition de services nommée PA*X* (*Pattern-Based Approach for Composition of Services*). Cette approche se base sur la notion d'intention et de contexte afin de formaliser les besoins des utilisateurs et de représenter les aspects environnementaux (culturel, organisationnel, spatial, temporel) et non fonctionnels (qualitatif, sécuritaire) liés aux utilisateurs (rôle, compétence).

Dans leur travaux, ces auteurs proposent l'utilisation des *patrons* de compositions afin d'assurer l'accessibilité, la réutilisation et l'adaptabilité des compositions. Ces patrons réalisent une certaine *intention* dans un *contexte* donné. D'un côté, l'intention est structurée selon le modèle de Prat (Prat, 1997). De l'autre côté, le *contexte* est exprimé sous la forme d'*assertions contextuelles* exprimées sous forme de *type* (temporelle, financière, rôle, compétence, etc.) et de sa *formulation*. Ces deux notions sont basées sur les concepts de l'ontologie contextuelle proposée par (Ramadour et Fakhri, 2011). Les assertions contextuelles sont combinées à travers les connecteurs logiques AND et NOT.

Pour découvrir les compositions de services nécessaires pour répondre aux besoins des utilisateurs, Ramadour et Fakhri (Ramadour et Fakhri, 2011) se basent sur la notion de rapprochement entre l'intention et le contexte de l'utilisateur avec ceux des patrons disponibles. Ce rapprochement comporte essentiellement deux types de traitements : (i) la *similarité des intentions ;* et (ii) la *compatibilité de contexte*. Ces deux traitements se basent sur une ontologie linguistique. D'une part, la *similarité des intentions* repose sur la similarité sémantique entre les actions et les objets. Ainsi, deux intentions sont similaires si leurs *actions* (verbes) et leurs *objets* sont *sémantiquement équivalents* selon l'ontologie

linguistique. Cette équivalence comporte les liens d'hyponymie et de synonymie, et est évaluée soit à une mesure exacte, soit à un poids entre [0..1] reflétant l'équivalence sémantique. D'autre part, la *compatibilité de contexte* se base sur des assertions contextuelles. Les assertions contextuelles sont séparées en deux groupes. Le premier groupe comporte les assertions positives, celles qui sont combinées par l'opérateur AND. Le deuxième groupe comporte les assertions négatives, celles qui sont combinées par l'opérateur NOT. Ainsi, la compatibilité de deux contextes commence par comparer séparément les assertions positives et négatives. Cette comparaison se base sur l'équivalence sémantique, selon l'ontologie linguistique, entre le *type* et la *formulation* de deux assertions contextuelles.

La découverte des services composites représente un des trois opérateurs de l'approche proposée par Ramadour et Fakhri (Ramadour et Fakhri, 2011), en plus de l'opérateur de *spécialisation* qui permet de descendre dans le niveau d'abstraction et *d'opérationnalisation* qui permet d'assembler les compositions fournies par les patrons d'une manière abstraite. Ces opérateurs représentent le processus de manipulation des compositions de services, lequel prend comme entrée le besoin à satisfaire et comme sortie la composition de services retrouvée à l'aide du processus de découverte décrit ci-dessus.

3.5.2.4.3. Extension de WSMO pour la sensibilité au contexte

Afin de suivre l'évolution des applications mobiles, l'approche WSMO a été étendue afin de prendre en considération divers éléments non fonctionnels, tels que la notion de contexte (Grenon, 2009). L'incorporation des informations contextuelles dans WSMO a été traitée dans le cadre du projet européen SOA4ALL[2]. L'objectif principal de ce projet est de fournir un cadre global qui intègre les avancées technologiques (*i.e.* SOA, gestion de contexte, Web 2.0 et Web sémantique) dans une plateforme de prestation de services cohérente et indépendante du domaine. Dans ce cadre, Grenon (Grenon, 2009) propose des mécanismes et des directives pour spécifier des *dimensions* structurant et uniformisant les informations contextuelles. Ces dimensions fournissent un ensemble d'informations contextuelles pertinentes dont certaines on été introduites dans une extension minimale de WSMO. Cette extension repose d'abord sur la relation entre l'intention et ces dimensions, considérées comme aspects non fonctionnels de l'intention. Pour cela, une nouvelle classe d'éléments a été ajoutée à WSMO et spécialisée dans les relations ci-dessus. Enfin, une notion d'agent pour la collecte de l'information contextuelle a aussi été ajoutée.

Outre l'approche de Grenon (Grenon, 2009), Saadon et Mohamad (Saadon et Mohamad, 2011) propose également une extension de WSMO, appelée WSMO-M, destinée à l'Informatique Mobile. Celle-ci considère le contexte en tant que propriété non fonctionnelle. Ces auteurs considèrent que l'ensemble des propriétés non fonctionnelles (NFP – *Non-Functional Properties*) proposé par WSMO, tel que *description*, *relation*, *sécurité*, *exactitude*, *couverture*, *version*, *passage à l'échelle*, etc., ne sont pas suffisamment expressives et

[2] http://www.soa4all.eu/

flexibles pour prendre en considération l'attribut *contexte*. Par conséquence, et comme l'illustre la Figure 18, ils proposent une nouvelle catégorie de la classe NFP appelée *contexte.* Celle-ci contient toutes les informations contextuelles qui peuvent être associées à la classe *service Web,* telles que ressource, moyen de paiement, localisation et temps, ainsi qu'à la classe *intention*, telles que le profil de l'utilisateur et du dispositif, la localisation et le temps.

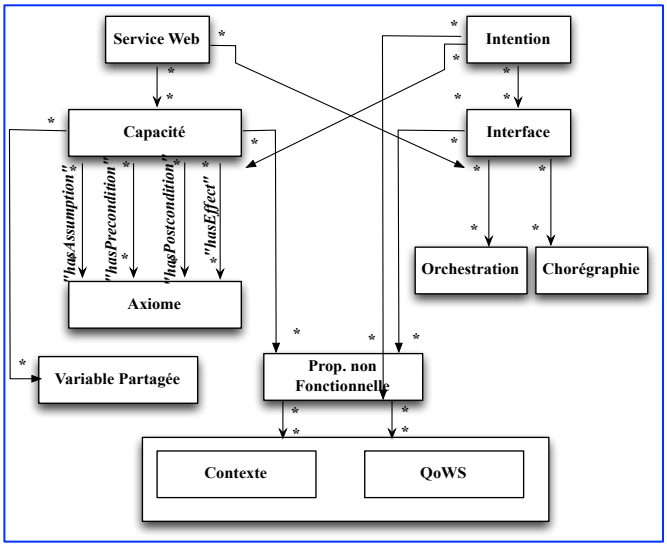


Figure 18. Extension des propriétés non fonctionnelles du modèle conceptuel de WSMO (d'après (Saadon et Mohamad, 2011))

De plus, ces auteurs proposent un mécanisme de découverte de services qui repose sur une *mise en correspondance des propriétés fonctionnelles* (pré-conditions et post-conditions), mais également des *éléments de contexte* liés au service et à l'utilisateur et une mise en *correspondance sémantique* entre l'intention qu'un service permet de satisfaire et la requête de l'utilisateur exprimée sous la forme d'intention. Celle-ci repose sur les mêmes degrés de mise en correspondance proposés par (Paolucci et al., 2002) : *exact*, *plug-in*, *subsume* et *fail*.

Les approches précédemment citées visent la satisfaction immédiate d'une requête directe de l'utilisateur. Il s'agit d'un comportement réactif et non proactif. Dans la section suivante, nous présentons ces approches proactives, de la prédiction de services, lesquelles représentent le deuxième challenge de SIP auquel nous nous intéressons.

3.5.3. La prédiction de services

A ce jour, une majorité de systèmes sensibles au contexte sont simplement réactifs, prenant les décisions en se basant seulement sur le contexte courant. Les recherches dans les systèmes anticipatoires et proactifs, notamment par la prédiction de la situation future de l'utilisateur, sont encore à leurs débuts. Dans de tels systèmes, un utilisateur peut avoir un ensemble d'habitudes. En exploitant celui-ci, nous pouvons améliorer la transparence des systèmes en réduisant l'effort de compréhension de l'utilisateur par l'anticipation de ces besoins. Plusieurs approches ont été proposées afin d'atteindre ce caractère anticipatoire soit des *services*, par la *recommandation* (Abbar et al., 2009) (Xiao et al., 2010)(Yu et al., 2012), soit du *contexte*

d'utilisation, par la *prédiction (Meiners et al., 2010)(Boytsov et Zaslavsky, 2011)*. Les prochaines sections introduisent certaines de ces approches.

3.5.3.1. Prédiction de contexte

Dans l'Informatique Pervasive, plusieurs recherches ont été menées dans le cadre de la prédiction de contexte d'utilisation. Ces contributions visent à introduire de nouvelles techniques de prédiction afin d'augmenter le caractère dynamique des systèmes pervasifs.

3.5.3.1.1. L'approche proposée par Mayrhofer et al.

Une des premières contributions dans la prédiction de contexte a été proposée par Mayrhofer (Mayrhofer, 2004). Cet auteur propose une architecture et un cadre pour la prédiction de contexte. Il est basé sur une classification non supervisée, qui tente de trouver des *clusters* de contexte, jusque-là inconnues à partir des données d'entrée, représentant des patrons récurrents. Cette approche interprète le contexte comme des états, dans lesquels un utilisateur ou un dispositif avance d'un état à un autre. Ainsi, l'interprétation d'un changement de contexte, comme une trajectoire d'état, permet de prévoir le développement futur de la trajectoire, et par conséquent prédire le contexte attendu.

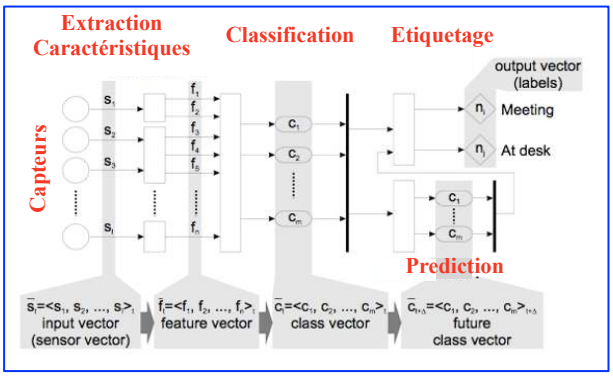


Figure 19. Architecture de prédiction de contexte (d'après (Mayrhofer, 2004))

Mayrhofer (Mayrhofer, 2004) propose un processus en cinq étapes illustré à la Figure 19, qui prend en entrée des séries temporelles (ensembles d'observations, chacune enregistrée à un temps précis) et qui offre en sortie le contexte courant de l'utilisateur (ou d'un dispositif) ainsi que son contexte futur prédis.

L'étape d'***acquisition de contexte*** se charge de collecter les informations contextuelles à partir d'un ensemble de capteurs physiques ou logiques. Les données ainsi capturées sont transformées en des caractéristiques *(features)* plus significatives lors de l'étape d'***extraction des caractéristiques*** afin de mieux les interpréter. Ensuite l'étape de ***classification*** se charge de reconnaître les patrons récurrents, appelés *clusters*, dans l'espace des caractéristiques définies lors de l'étape précédente. Cette approche de classification utilise un vecteur de caractéristiques lequel peut éventuellement être affecté à plusieurs *clusters* avec certains

degrés d'appartenance. Ce degré représente la probabilité que le vecteur de caractéristiques appartienne à un *cluster*. Cette étape repose sur une extension de l'algorithme de classification LLGNG (*Lifelong Growing Neural Gas*) (Hamker, 2001). Enfin, l'***étiquetage*** affecte des noms descriptifs aux *clusters* individuels ou aux combinaisons de *clusters*. Une fois les *clusters* identifiés, l'étape de **prédiction** va essayer de prédire le *cluster* de contexte futur sur la base de l'historique observé. Cette étape s'appuie sur le vecteur de *clusters* généré par l'étape de classification. Cela permet de prévoir plus qu'une futur*e meilleure correspondance* de contexte en exploitant les degrés appartenance aux *clusters*. L'objectif est de générer des vecteurs de *cluster* pour des points dans le futur, qui correspondent au vecteur de *cluster* courant fourni par l'étape de classification. Cela permet d'alimenter les vecteurs de *clusters* prévus (prédis) dans l'étape d'étiquetage pour fournir des étiquettes de contextes prévus pour une utilisation dans des applications dynamiques.

3.5.3.1.2. L'approche proposée par Sigg et al.

Semblable à Mayrhofer (Mayrhofer, 2004), Sigg et al. (Sigg, 2008) (Sigg et al., 2010) proposent une définition formelle de la tâche de prédiction de contexte répondant à la problématique posée sur la qualité de contexte et sur la prise en compte de l'ambiguïté de cette information qui peut être incomplète (*cf.* section 2.3.4.1). Ils proposent une architecture de prédiction de contexte basée sur une méthode d'alignement, à partir de laquelle les informations de contexte manquantes sont déduites.

L'architecture proposée par Sigg et al. (Sigg et al., 2010), à l'instar de Mayrhofer (Mayrhofer, 2004), est aussi basée sur des patrons de contexte type que l'algorithme d'apprentissage construit pour guider le module de prédiction. Ceux-ci sont composés de séries temporelles auxquelles on attribue des poids décrivant l'importance attachée à chacune. Ainsi, si la série temporelle observée est identique ou similaire à un patron de contexte, alors le poids attaché à l'alignement dans la base est renforcé. Dans le cas contraire, un nouvel alignement est rajouté à la base et les poids des autres patrons de contexte qui sont différents du nouveau patron de contexte observé sont réduits. Ces séries temporelles représentent l'historique de contexte. Elles sont alimentées par toutes les sources de contexte disponibles dans la couche d'acquisition de contexte de l'architecture.

Le *module de* ***prédiction de contexte*** se base sur une méthode d'*alignement*, permettant de prédire la continuation la plus probable d'une série temporelle à partir du suffixe de la séquence observée. Cette approche de prédiction par alignement cherche à aligner deux chaînes à l'aide de leurs écarts de sorte que le nombre de positions correspondant aux deux chaînes soit maximisé. Enfin Sigg et al. (Sigg et al., 2010) proposent également un *module d'****apprentissage*** continu dans le but de s'adapter à l'évolution des environnements ou des habitudes des utilisateurs. Il surveille en permanence les séries temporelles enregistrées dans l'historique de contexte et met à jour les patrons enregistrés. Cependant, dans le cadre de ces travaux, ces auteurs ne proposent aucune implémentation spécifique pour ce module d'apprentissage. Seules les conditions requises pour celui-ci sont indiquées, notamment

l'interface spécifiée par l'historique de contexte et le langage de description des règles, représentant les patrons.

3.5.3.1.3. L'approche SCP

Meiners et al. (Meiners et al., 2010) proposent une approche de prédiction de contexte nommée SCP (*Structured Context Prediction*). Celle-ci est basée sur deux principes clés. Le premier repose sur l'utilisation des connaissances du domaine d'application que les développeurs peuvent intégrer au moment du design. Ces connaissances sont décrites selon un *modèle de prédiction* qui spécifie la manière dont les prédictions doivent être exécutées et qui configure le système de prédiction. Le deuxième principe expose l'application de plusieurs méthodes de prédiction, qui sont échangeables. Ces méthodes sont proposées afin d'assurer l'exactitude et l'efficacité des prédictions spécifiques à un domaine. Elles peuvent être choisies et combinées par les développeurs d'applications. D'après (Meiners et al., 2010), le *modèle de prédiction* attribue une méthode pour chaque variable afin de prédire sa valeur. La méthode utilise comme entrée les valeurs d'autres variables qui sont soit prédites par leurs propres méthodes, soit connues (mesurables par des capteurs).

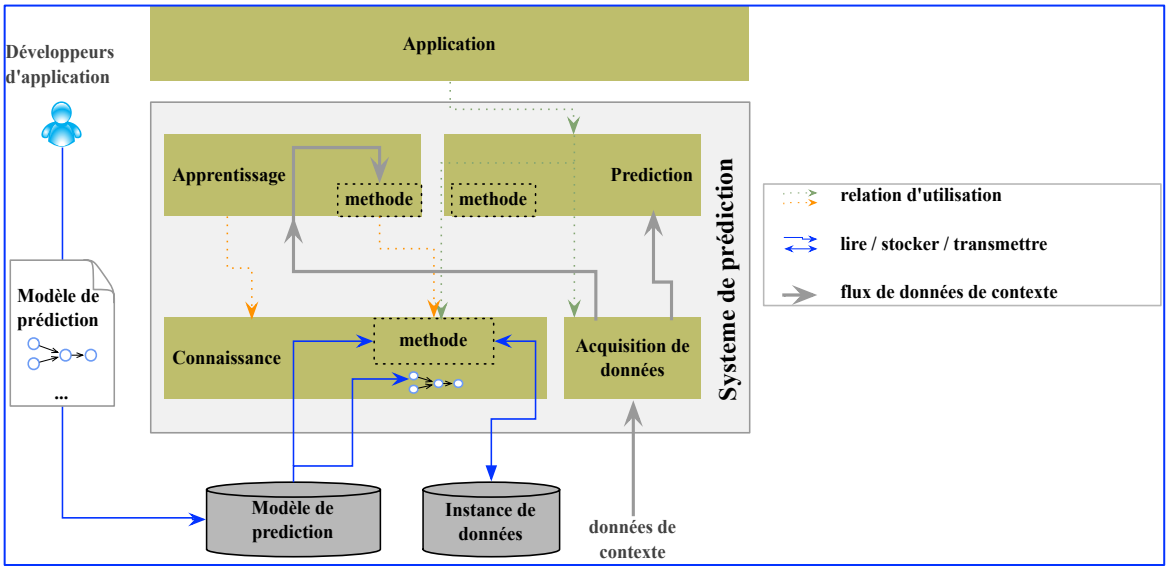


Figure 20. Architecture du système de prédiction selon l'approche SCP (d'après (Meiners et al., 2010))

De plus, ces auteurs proposent une architecture pour un système de prédiction qui peut être utilisé comme un composant réutilisable par des applications sensibles au contexte. Cette architecture, comme l'illustre la Figure 20, se base sur quatre éléments principaux. Le premier représente le *composant de connaissances*, lequel contient les connaissances sur les relations, les caractéristiques et les régularités qui déterminent le contexte. Ce composant est constitué d'un modèle de prédiction et des instances de données. Ensuite, cette architecture comporte le *composant d'acquisition de données* qui se charge d'acquérir des données de contexte. Un troisième composant, celui *de l'apprentissage,* obtient régulièrement des données de contexte du composant d'acquisition et les affecte aux méthodes de prédictions comme des données d'apprentissage. Ensuite, ce composant crée et met à jour les connaissances en utilisant les

résultats de l'exécution de ces méthodes. Le dernier composant représente celui *de la prédiction.* Il utilise les connaissances acquises pour la prédiction effectuée sur demande. Ce composant repose sur un algorithme proposé par (Meiners et al., 2010) qui se charge de coordonner l'ensemble de méthodes. Cet algorithme est inspiré de l'algorithme « *Stochastic Simulation »* (Jensen, 2001) initialement développé pour les réseaux bayésiens.

3.5.3.2. Systèmes de recommandation centrés sur le contexte

Dans le cadre de la prédiction de services, nous avons souligné, à travers la littérature, des contributions dans le domaine de la recommandation de services. Ces contributions sont similaires aux approches de prédiction puisqu'elles cherchent à proposer les prochains services qui peuvent intéresser l'utilisateur et répondre à son besoin futur.

3.5.3.2.1. L'approche proposée par Abbar et al.

Abbar et al. (Abbar et al., 2009) fournissent une approche, dans laquelle les services sont recommandés sur la base des fichiers *log* de l'utilisateur et de son contexte courant. Ces auteurs proposent un système de recommandation sensible au contexte nommé CARS – (*Context-Aware Recommender System*), qui se base sur le *profil de l'utilisateur* et sur son *contexte*. Cette approche utilise une architecture, sur laquelle un ensemble de concepts génériques (profil de l'utilisateur, utilisateur actif, contexte, contexte actif, profil contextualisé et profil opérationnel) et de services de personnalisation (découverte de services, contextualisation des services, service de liaison, service de mise en correspondance) sont déployés afin de rendre une application sensible au contexte.

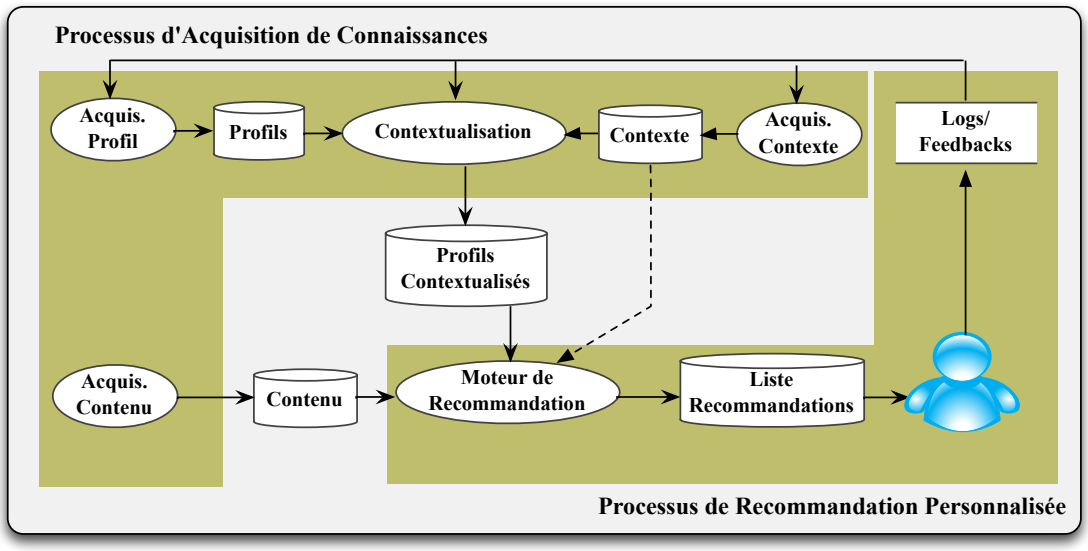


Figure 21. Architecture du système de recommandation sensible au contexte (d'après (Abbar et al., 2009))

Cette architecture, comme l'illustre la Figure 21, se répartie en deux processus. Le premier est le *processus d'acquisition de la connaissance* qui se charge d'acquérir et de gérer les connaissances (profil de l'utilisateur, description du contenu et du contexte) dont CARS a

besoin pour la recommandation des services. Dans ce processus, l'acquisition de contexte se base sur les logs contenant un ensemble d'enregistrements lesquels contiennent à leur tour les informations contextuelles (date, temps, dispositif, etc.). Ces enregistrements sont regroupés en un ensemble de *clusters* qui représentent le contexte régulier. Ce processus utilise d'abord l'algorithme *« Agglomerative Hierarchical Clustering »* (AHC) pour estimer le nombre de *clusters*. Le nombre de *clusters* retrouvés ainsi que les enregistrements représentent, par la suite, l'entrée de l'algorithme *« K-means »* utilisée pour déterminer les *clusters*. De plus, ce processus inclut une étape de contextualisation du profil de l'utilisateur laquelle se charge de découvrir les relations entre les éléments de contexte et les profils de l'utilisateur.

Le deuxième processus représente le *processus de recommandation personnalisée* proprement parlé. Le moteur de recommandation accepte comme entrée le profil contextualisé de l'utilisateur ainsi que son contexte courant. Au cours de ce processus, un algorithme appelé *« Top Contextual K Neighbors »* est appliqué afin de déterminer un nombre « K » d'utilisateurs qui sont les plus similaires à l'utilisateur actif par rapport à son profil. Ensuite, dès que les K plus proches des voisins sont déterminés, les classements attribués aux éléments à recommander sont agrégés. Le résultat d'agrégation permet de décider s'il est pertinent ou pas de recommander un certain item représentant le service à recommander à l'utilisateur.

Comme pour les autres approches de prédiction de contexte, cette approche nécessite, afin de sélectionner et de recommander des services, des données historiques (log), qui ne sont pas toujours disponibles. En effet, elle a besoin d'une première phase de collecte afin d'obtenir suffisamment de données qui vont être traitées par la suite. De ce fait, les premières recommandations de services peuvent être biaisées et surtout peuvent ne pas intéresser l'utilisateur puisqu'elles ne sont pas déduites à partir de ses habitudes.

3.5.3.2.2. Approche proposée par Xiao et al.

A l'encontre de l'approche précédente, l'approche proposée par (Xiao et al., 2010) propose une procédure de recommandation de services sans compter sur des fichiers logs. Ces auteurs (Xiao et al., 2010) proposent une approche qui détermine dynamiquement un modèle de contexte. Ce modèle gère divers types et valeurs contextuels, et recommande ensuite des services en utilisant ces informations. Cette approche utilise les ontologies pour améliorer la sémantique des valeurs de contexte associées à un utilisateur et identifie automatiquement les relations entre ces différentes valeurs. Ces auteurs se basent sur les relations entre les valeurs de contexte afin de trouver les services potentiels dont l'utilisateur pourrait avoir besoin. Une relation définit la façon dont les classes ou les individus peuvent être associés entre eux dans une ontologie. Elle peut être soit une relation prédéfinie par l'ontologie (*subclass*, *partOf*, *complement* ou *equivalence*), soit une relation spécifique représentée par les propriétés de l'ontologie. Au lieu de définir manuellement les règles de type *si-alors* en utilisant des types de contexte spécifiques ou des valeurs prédéfinies, cette approche utilise les relations entre les valeurs de contexte afin de déduire le contexte dans lequel émerge les besoins des utilisateurs.

La Figure 22 présente un aperçu de cette approche. Pour identifier les relations, une recherche des ontologies disponibles pour étendre la sémantique des valeurs de contexte est effectuée. Les relations identifiées sont utilisées afin de découvrir le contexte dans lequel les besoins des utilisateurs sont exprimés. Ceci conduit à la génération des critères de recherche des services correspondants. Ces critères sont utilisés à leur tour pour la recherche de nouveaux services, permettant au système de recommander ces services aux utilisateurs.

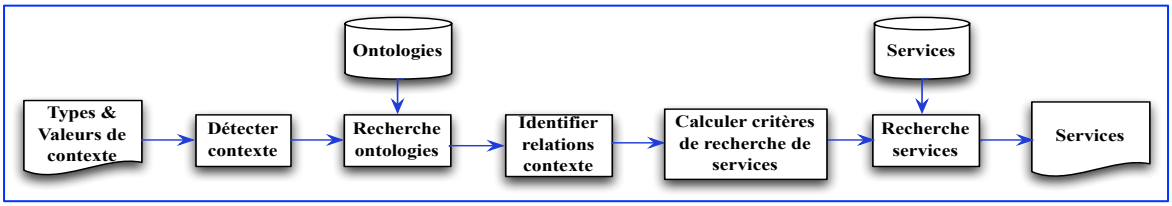


Figure 22. Les étapes de l'approche de recommandation de services sensibles au contexte (Xiao et al., 2010)

Les relations de contexte sont de deux types, les relations entre deux valeurs de contexte et les relations entre multiples valeurs. Les *relations entre deux valeurs de contexte* identifient deux entités appartenant à des ontologies différentes, en se basant sur la *similarité* entre elles. Cette similarité peut être *entre deux propriétés atomiques* (si elles ont le même nom et la même valeur de propriété), ou *entre deux classes ou deux propriétés non atomiques* (si elles ont le même nom ou n'importe quelle propriété définie dans l'une est similaire à une propriété dans l'autre). En se basant sur cette similarité, quatre relations peuvent être identifiées entre deux valeurs de contexte, à savoir : *intersection*, *complémentaire*, *équivalence* et *indépendance*. Les *relations entre multiples valeurs de contexte* sont décrites à l'aide du modèle E-R (*Entity Relationship*). Pour chaque relation entre deux valeurs, les valeurs sont converties en entités du modèle et les types de relations en nœuds de relation dans le modèle.

Concernant la génération des critères de recherche, cette approche définie des règles génériques pour inférer les besoins des utilisateurs à partir du modèle E-R. Ensuite, elle extrait les critères de recherche (mots clés) de la description des besoins des utilisateurs pour rechercher les services à recommander.

3.6. Conclusion et considérations finales

Dans ce deuxième chapitre de l'état de l'art, nous avons introduit la notion de *service*. Ces services se caractérisent essentiellement par leur i*ndépendance par rapport aux aspects technologiques et à leur implémentation* et ont été conçue principalement pour répondre à des besoins de réutilisation et d'interopérabilité (Papazoglou et Georgakopoulos, 2003). Dans le cadre des systèmes sensibles au contexte, multiples travaux (Maamar et al., 2006) (Baldauf et al., 2007) (Toninelli et al., 2008) (Preuveneers et al., 2009) démontrent l'intérêt de la notion de service pour l'adaptabilité d'un système. La notion de service permet ici de masquer l'hétérogénéité technologique des environnements pervasifs. Ainsi, l'orientation service permet ainsi de répondre au besoin de gestion de l'hétérogénéité des SIP et notamment des actions que ces systèmes pourront proposer pour satisfaire les besoins des utilisateurs.

Nous avons également souligné un ensemble de challenges auxquels les SIP orientés services doivent faire face. Ces challenges représentent les éléments clés pour la construction d'un SIP proactifs et réactifs aux changements de l'environnement et aux besoins des utilisateurs. Nous nous sommes ainsi focalisés sur d'eux d'entre eux, à savoir *la découverte et la prédiction de services*. Nous avons ainsi exposé une multitude d'approches de découverte et de prédiction de services. Avec l'apparition du Web sémantique, différentes approches de découverte de services sémantiques, telles que (Paolucci et al., 2002) (Sycara et al., 2003) (Klusch et al., 2009), ont été proposées. Celles-ci, même si elles sont prometteuses, demeurent assez limitées quant à son emploi dans le cadre d'un environnement pervasif et donc dans les SIP. En effet, ces approches se limitent aux capacités des services en termes d'entrée et de sortie et ne prennent pas en considération le caractère dynamique de l'environnement par la prise en compte de l'information contextuelle. De plus, elles ne tiennent pas compte du besoin réel de l'utilisateur derrière sa demande d'un service.

Les approches suivantes même si elles sont différentes, représentent une évolution des SI vers des SI centrés utilisateur, soit par une approche intentionnelle (Mirbel et Crescenzo, 2010b) (Aljoumaa et al., 2011) (Olsson et al., 2011), soit par une approche contextuelle (Suraci et al., 2007) (Toninelli et al., 2008) (Vanrompay et al., 2011). Cependant, les approches sensibles au contexte requièrent des connaissances techniques complexes de l'utilisateur s'il veut comprendre le choix suggéré, alors que normalement ces utilisateurs demandent simplement un service qui répond à leurs besoins. Cette problématique, est évoquée par les approches intentionnelles. Celles-ci focalisent sur les besoins des utilisateurs décrits sous forme d'intention. Toutefois, la satisfaction de ces intentions par les services peut varier selon le contexte dans lequel se situe l'utilisateur. Ces approches ne prennent pas en considération cet aspect, ce qui peut nuire au résultat proposé à l'utilisateur en lui offrant un service mal adapté à son contexte d'usage.

Nous pensons que ces approches sont en réalité complémentaires, et qu'une telle évolution ne peut être atteinte véritablement que par la combinaison de ces deux approches. À notre avis, seulement un mécanisme de découverte de services basé à la fois sur le contexte et sur l'intention de l'utilisateur est en mesure de répondre à des questions telles que « *pourquoi un service est utile dans un contexte donné ?* » ou « *dans quelles circonstances émerge le besoin d'un service ?* ». Cette vision a commencé à se développer dans la littérature. Quelques auteurs, dont (Santos et al., 2009) (Ramadour et Fakhri, 2011) (Ma et al., 2011), proposent déjà d'associer ces deux notions lors de la découverte et de la composition de services. Cependant, pour beaucoup d'entre eux, cette association reste assez floue. Le contexte reste souvent confiné à un rôle de filtre pour la découverte de services, étant présenté comme une partie des entrées nécessaires aux services, alors que les intentions sont vues comme de simples étiquettes permettant de relier les demandes des utilisateurs aux services. Nous croyons, au contraire, que le contexte ne peut être réduit à de simples paramètres d'entrées ou de sorties. Non seulement il influence l'exécution du service, mais il caractérise le service lui-même et les intentions affichées par le service.

Selon notre analyse, aucun des travaux cités ne propose un mécanisme de découverte de services qui combine et exploite réellement le contexte et l'intention. Un tel mécanisme de découverte de services est essentiel dans le cadre d'un SIP transparent et centrée utilisateur, lequel doit se caractériser par son adaptabilité au contexte et sa compréhension de l'utilisateur et de ses besoins. Le même constat peut être fait pour les mécanismes de prédiction de services. Nous avons souligné deux familles d'approches intéressantes, à savoir les approches de prédiction de contexte et les approches de recommandation de services. Les approches de *prédiction de contexte* dans les environnements pervasifs, tels que (Mayrhofer, 2004) (Sigg, 2008) (Meiners et al., 2010), tentent de prédire le prochain contexte de l'utilisateur en fonction de son contexte courant et de son historique. Tous ces auteurs proposent des architectures et des méthodes intéressantes qui assurent la prédiction de contexte de haut et de bas niveau. Cependant, aucun de ces travaux n'associent à ce contexte les services ou les activités qu'un utilisateur invoque. Ils se focalisent uniquement sur la prédiction de contexte suivant ou la prédiction de la continuité de celui-ci, sans pour autant souligner comment ce contexte prédit est utilisé pour découvrir ou prédire le service.

A l'opposé les approches de *recommandation de services selon le contexte*, tels que (Abbar et al., 2009) et (Xiao et al., 2010), proposent des mécanismes pour recommander le service suivant selon le contexte de l'utilisateur. La plupart de ces approches tiennent compte uniquement des informations contextuelles, sans prendre en considération les besoins réels de l'utilisateur derrière un service, c'est-à-dire, de ses objectifs. Ils proposent une mise en œuvre à l'utilisateur, en ignorant pourquoi celle-ci est nécessaire.

Aujourd'hui, un enjeu important dans le domaine des SIP est de se positionner au niveau de l'utilisateur et de prendre en considération sa mobilité et la dynamique de l'environnement qui l'entoure. Les approches que nous avons présentées, que ce soit la découverte ou la prédiction de services, n'exploitent pas vraiment l'étroite relation entre la notion d'intention, qui représente le besoin de l'utilisateur, et la notion de contexte. En conséquence, plusieurs possibilités sont offertes à l'utilisateur, qui n'est pas toujours en mesure de les utiliser proprement, ni de comprendre ce qui lui est proposé. Afin de répondre aux exigences des SIP, il nous semble nécessaire de concevoir et de développer des mécanismes de découverte et de prédiction de services qui doivent être guidés à la fois par le contexte et par l'intention.

Chapitre 4. VISION INTENTIONNELLE ET CONTEXTUELLE DES SYSTEMES D'INFORMATION PERVASIFS

4.1. INTRODUCTION

Après une analyse de la littérature (*cf.* Chapitre 2 et Chapitre 3), nous pouvons constater que les SI traditionnels ne sont plus adaptés à l'environnement pervasif dans lequel évolue l'utilisateur actuellement. Aucun des éléments caractérisant un environnement pervasif (*hétérogénéité*, *dynamisme, etc.*) n'est particulièrement pris en compte dans ces systèmes, conçus pour les environnements de bureau stables et contrôlés. Ceci à fait apparaître une nouvelle génération des SI, les *Systèmes d'Information Pervasifs*.

Nous proposons dans cette thèse une nouvelle vision intentionnelle et contextuelle pour la gestion de ces nouveaux Systèmes d'Information Pervasifs orientés services. Cette vision répond aux problématiques de *transparence*, d'*adaptation à l'environnement* et d'*adaptation aux utilisateurs* relevés dans le premier chapitre (*cf.* section 1.2). Notre vision est centrée utilisateur, qui devient le centre des nouveaux Systèmes d'Information Pervasifs.

Ce chapitre rappelle notre contexte de recherche, à savoir les SIP, et notre problématique de départ, laquelle a conduit à notre proposition d'une vision intentionnelle et contextuelle des SIP. Nous présentons ensuite un aperçu détaillé de la solution répondant à la problématique et la mise en place des hypothèses en pratique.

4.2. RAPPEL DU CONTEXTE DE RECHERCHE ET DE LA PROBLEMATIQUE

Dans cette section, nous rappelons notre contexte de recherche et la problématique qui a conduit à notre proposition (*cf.* Chapitre 1).

4.2.1. Contexte de recherche

Comme nous l'avons présenté dans le Chapitre 2, l'Informatique Pervasive est devenue une réalité grâce à l'intégration transparente de plusieurs périphériques dans notre vie quotidienne. Les nouvelles technologies ont élargi les frontières des Systèmes d'Information (SI) en dehors de l'environnement des entreprises. Le BYOD (*Bring Your Own Device*) illustre assez bien cette tendance : les employés apportent leurs propres dispositifs au bureau et continuent à les utiliser pour accéder au SI, même quand ils sont en déplacement. La conséquence de cette évolution technologique est que les SI doivent maintenant faire face à un environnement pervasif, et à l'avenir, intégrer des éléments physiques ainsi que logiques et

organisationnels. Par ailleurs, dans les dernières années, les SI ont massivement adoptés une approche basée sur les services, devenant ainsi des systèmes orientés services exposant ses fonctionnalités en tant que services.

Ainsi, émerge une nouvelle génération de Systèmes d'Information, appelée les Systèmes d'Information Pervasifs (SIP). Les SIP ont l'intention d'améliorer la productivité de l'utilisateur en permettant aux services d'un SI d'être disponibles à tout moment et à n'importe quel endroit. Ces systèmes déplacent le paradigme d'interaction de l'informatique de bureau aux nouvelles technologies, passant d'un environnement entièrement contrôlé (le bureau) vers un environnement pervasif hautement dynamique.

4.2.2. Problématique

Les Systèmes d'Information Pervasifs doivent faire face à des environnements pervasifs, sans laisser derrière le fait qu'ils demeurent des Systèmes d'Information. Les SIP doivent faire face à l'*hétérogénéité* qui caractérise les environnements pervasifs. La *transparence* est donc nécessaire afin de cacher aux utilisateurs cette hétérogénéité des dispositifs, des infrastructures et des services. Cette transparence est d'autant plus nécessaire en raison du rôle central que jouent les SI dans les entreprises. Ces systèmes sont conçus pour aider les utilisateurs à atteindre des objectifs métiers bien précis. Par conséquent, lors de l'utilisation de tels systèmes, les utilisateurs doivent se concentrer sur leurs propres tâches/activités et non sur la technologie elle-même. Sans transparence, tout Système d'Information Pervasif ne sera pas en mesure de remplir avec succès son rôle de SI.

Au cours de la dernière décennie, beaucoup de recherches ont été effectuées sur les systèmes et principalement sur les services sensibles au contexte (Maamar et al., 2006) (Toninelli et al., 2008) (Vanrompay et al., 2011) (Bronsted et al., 2010) (Truong et Dustdar, 2009). Ces travaux de recherche proposent un comportement sensible au contexte pour la découverte et la composition de services. La sensibilité au contexte devient un élément clé pour soutenir de tels environnements pervasifs. Ainsi, les Systèmes d'Information Pervasifs devraient fournir des capacités de sensibilité au contexte afin de faire face aux changements dynamiques de l'environnement et d'améliorer l'efficacité de l'utilisateur.

Néanmoins, les SIP, contrairement aux systèmes pervasifs, doivent aussi se comporter comme des Systèmes d'Information traditionnels, gérant les services en fonction de l'utilisateur et des objectifs métiers. Les SIP représentent la prochaine génération des SI et ils doivent aussi faire face à ce rôle de SI. En raison de leur rôle stratégique, les SIP ne peuvent pas être conçus comme des systèmes pervasifs « normaux ». Les SIP doivent être « contrôlables » et « maîtrisables ». En d'autres termes, ils doivent être gérés et contrôlés par la Direction des Systèmes d'Information (DSI), puisqu'une exposition inappropriée d'un service interne peut avoir des conséquences importantes pour l'activité de l'entreprise. Ainsi, les comportements exploratoires et opportunistes, tels que ceux proposés par (Preuveneers et Berbers, 2010) et (Khan, 2010) ne peuvent pas être pleinement acceptés par la DSI. Ils représentent un risque pour le SI vu le rôle qu'il joue dans l'entreprise.

La conception des Systèmes d'Information Pervasifs qui réponde aux exigences des SI et des systèmes pervasifs est un défi. Les SIP sont censés être conçus pour améliorer l'efficacité de l'utilisateur en tenant compte de l'environnement pervasif dans lequel l'utilisateur émerge. Pour réussir, les SIP doivent tirer profit des opportunités offertes par les environnements pervasifs, et notamment la mobilité de l'utilisateur, tout en offrant aux utilisateurs des services du Système d'Information de manière transparente. Les SIP doivent être conçus afin de fournir les services les plus appropriés que les utilisateurs ont besoin pour satisfaire leurs objectifs dans leur situation courante.

Pour récapituler, nous résumons notre problématique à un *problème de conception et de réalisation d'un SIP transparent, s'adaptant à l'environnement (sensibilité au contexte) et à l'utilisateur (prise en compte des intentions et de la mobilité de l'utilisateur)*. A l'heure actuelle, il existe peu de modèles ou de méthodes permettant aux concepteurs et aux développeurs de prendre en compte ces besoins lors de la conception d'un SIP.

Ainsi, nous avons privilégié l'axe conceptualisation du système à développer pour élaborer notre solution, à l'instar de Mathieu Petit (Petit, 2010), qui propose un cadre conceptuel pour les Systèmes d'Information mobiles et distribués (*cf.* section 3.5.2.2.5) exploitant des modèles géographiques. La cible du système à développer, dans notre cas, ne prend pas en considération uniquement l'aspect géographique mais s'étend plus généralement à l'adaptation à l'environnement et à l'utilisateur.

4.3. APERÇU DE LA SOLUTION

La conception de Systèmes d'Information Pervasifs est un défi pour lequel la DSI n'a aucune aide. Nous soutenons que l'utilisateur doit être au centre de cette nouvelle génération de Systèmes d'Information, étant donné que ces systèmes doivent être conçus pour aider l'utilisateur à mieux satisfaire ses objectifs en fonction de l'environnement dans lequel il se trouve. En outre, cette vision devrait examiner tous les aspects des nouveaux SIP : leur besoin de transparence, l'hétérogénéité et la dynamique des environnements pervasifs, ainsi que les intentions qu'ils doivent satisfaire à partir du point de vue SI.

Ainsi, nous proposons notre vision des Systèmes d'Information Pervasifs. Après notre analyse de la littérature dans le Chapitre 2 et Chapitre 3, nous avons constaté que *l'orientation service,* la *sensibilité au contexte*, ainsi que l'*approche intentionnelle* représentent des approches très prometteuses à prendre en considération afin de concevoir un SIP transparent, non intrusif et compréhensible à l'utilisateur. Ces approches représentent des solutions pour résoudre les problèmes que nous avons soulevés ci-dessus.

Les prochaines sections résument notre vision intentionnelle et contextuelle des SIP et les différents éléments composant notre solution.

4.3.1. Notre vision intentionnelle et contextuelle des SIP : couplage entre services, contexte et intention

Notre vision des Système d'Information Pervasif est basée sur les notions d'intention, de contexte et de services. Elle représente notre solution pour répondre à la problématique de recherche soulevée et pour mettre en place les hypothèses, énumérées dans la section 1.3, en pratique. Cette solution représente une vision centrée sur l'utilisateur des SIP. Elle permet, d'une part, de se focaliser plus sur les besoins réels des utilisateurs à travers une approche intentionnelle, assurant ainsi une meilleure compréhension du système des objectifs de l'utilisateur derrière sa demande de services. Elle permet, d'autre part, de gérer l'hétérogénéité et la dynamique de l'environnement pervasif à travers une approche contextuelle. En effet, nous considérons les SIP et leurs éléments à la fois sous l'angle des SI et celui des environnements pervasifs, en observant leurs besoins respectifs de *contrôle*, d'*intentionnalité* et de *sensibilité au contexte*. Ceci est dans la perspective d'assurer la transparence nécessaire pour la conception d'un SIP.

Notre vision se caractérise par son *orientation service*. Selon notre analyse de l'état de l'art (*cf.* section Chapitre 3), nous avons soulevé la caractéristique principale de la notion de service, à savoir son *indépendance par rapport aux aspects technologiques et à leur implémentation*. Ainsi, une orientation service va permettre de répondre au besoin de gestion de l'hétérogénéité technique de l'environnement dans lequel évoluent les SIP et des actions que le système propose afin de satisfaire les besoins des utilisateurs. Dans notre vision, la notion de *service* permet de masquer l'hétérogénéité technologique des environnements pervasifs. Le concept « *service* » a été conçu pour répondre à des besoins de réutilisation et d'interopérabilité (Papazoglou et Georgakopoulos, 2003). Ses multiples usages dans les systèmes sensibles au contexte (*cf.* section Chapitre 3) démontrent également l'intérêt de la notion de service pour l'adaptabilité d'un système.

De plus, afin de mieux gérer l'hétérogénéité et la dynamique qui caractérisent l'environnement pervasif, notre vision se base sur une *orientation contexte.* Ceci va permettre d'adapter les SIP au contexte de l'utilisateur et à l'environnement. A partir de notre analyse de la littérature, nous avons soulevé le rôle central que joue cette notion de contexte dans les systèmes sensibles au contexte (*cf.* Chapitre 2). Notre approche présente les SIP comme des systèmes sensibles au contexte, qui s'adaptent aux changements sans pour autant demander à l'utilisateur de s'adapter lui-même à l'environnement.

Enfin, et dans la perspective de répondre au mieux aux exigences des utilisateurs et de se situer à leur niveau, notre vision des SIP est *orientée intention*. Ainsi, les SIP peuvent, d'une part, mieux comprendre les besoins des utilisateurs, et d'autre part, répondre d'une façon plus appropriée à ces besoins. Comme nous l'avons présentée dans la littérature (*cf.* section 3.4.3.1), la notion d'intention formalise, en général, les besoins de l'utilisateur. Une intention peut être considérée comme le but que nous voulons atteindre sans dire comment l'exécuter (Kaabi et Souveyet, 2007). Dans le cadre de notre vision, la notion d'intention est nécessaire afin que le système puisse mieux comprendre les besoins des utilisateurs et donc de répondre

à ses besoins de la manière la plus appropriée. En effet, l'approche intentionnelle nous permet de considérer le service du point de vue des exigences de l'utilisateur, en se concentrant sur *pourquoi* un service est nécessaire, et pas seulement sur *comment* il est exécuté. En fait, nous considérons que l'utilisateur ne nécessite pas un service uniquement parce qu'il est dans un contexte donné. Il requiert un service parce qu'il a une intention qu'un service peut satisfaire dans ce contexte d'usage (Najar et al., 2012a).

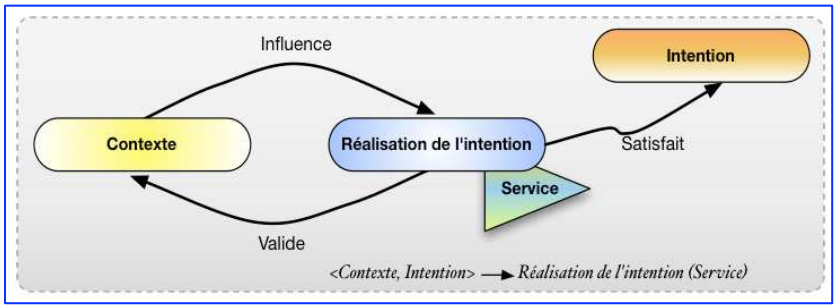


Figure 23. Contexte et intention dans l'orientation service

Par ailleurs, nous considérons que l'intention de l'utilisateur émerge dans un contexte donné et que ses réalisations ne sont valides que dans un contexte d'utilisation bien défini. Dans ce cadre, la notion de *contexte* représente un élément important dans le processus d'adaptation d'un système à l'utilisateur et à l'environnement, auquel nous souhaitons ajouter la notion d'intention. Ainsi, nous exploitons, dans notre vision des SIP, l'étroite relation entre les notions d'intention, de contexte et de service, illustrée à la Figure 23. Nous considérons que la satisfaction des intentions de l'utilisateur dans un SIP dépend du contexte dans lequel se trouve cet utilisateur. Pour nous, le contexte impacte directement la manière de satisfaire les intentions, et ainsi le choix des services qui seront exécutés.

Ainsi, en combinant les approches intentionnelles et contextuelles dans une orientation service et en exploitant la relation qui les lie, nous proposons une nouvelle vision centrée utilisateur d'un SIP transparent, non intrusif et compréhensible à l'utilisateur. Nous proposons, par la suite, des solutions pour mettre en œuvre cette vision intentionnelle et contextuelle des SIP centrés utilisateur.

4.3.2. Solution globale : de la conception à la mise en œuvre d'un SIP transparent et centré utilisateur

Nous proposons, dans le cadre de cette thèse, une solution globale pour mettre en place notre vision intentionnelle et contextuelle des SIP orientés services. Nous proposons celle-ci dans la perspective d'aider la DSI à concevoir un SIP transparent et centrée utilisateur. Nous commençons par introduire un *cadre conceptuel* des SIP appelé *« espace de services »* (*cf.* Chapitre 5) afin de conceptualiser le SIP et ses différents éléments. Nous proposons ensuite une *architecture de gestionnaire de SIP* (*cf.* Chapitre 9) qui interagit avec l'espace de services afin d'assurer les fonctionnalités de *découverte* (*cf.* Chapitre 7) et de *prédiction* (*cf.* Chapitre 8) des services pour l'utilisateur. Finalement, nous proposons une démarche méthodologique

de conception et de réalisation d'un SIP qui supporte le passage du cadre conceptuel vers les descriptions de services à l'intérieur du système.

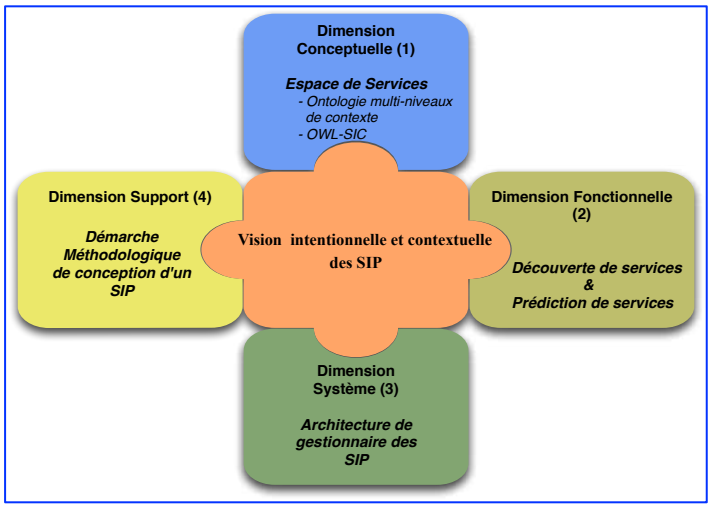


Figure 24. Vision intentionnelle et contextuelle des Systèmes d'Information Pervasifs

Nous proposons ainsi de présenter, comme l'illustre la Figure 24, notre solution globale selon quatre dimensions différentes : *conceptuelle*, *fonctionnelle*, *système* et *support*. Nous détaillons, dans les sections suivantes, chacune de ses dimensions.

4.3.2.1. Dimension conceptuelle : cadre conceptuel des SIP nommé « Espace de services »

Dans cette thèse, nous soutenons que la conception d'un SIP est devenue un problème difficile à gérer, en raison notamment de l'immersion des fonctionnalités d'un SIP dans un environnement pervasif. Il nous faut donc un cadre conceptuel permettant de masquer l'hétérogénéité de cet environnement et de spécifier les fonctionnalités qui seront proposées par le SIP et sous quelles conditions, et ceci de manière indépendante des technologies. La notion d'*espace de services* que nous proposons (*cf.* Chapitre 5) s'attaque à cette question.

Dans un espace de services s'intègrent, de manière transparente, l'ensemble des services offerts par le SI traditionnel (services logiques) et ceux offerts par l'environnement physique, grâce à l'intégration de différentes technologies dans cet environnement (services physiques), ainsi qu'un ensemble de capteurs qui renseignent l'utilisateur et le système sur cet environnement. La notion d'espace de services est un outil conceptuel permettant de gérer l'hétérogénéité des environnements pervasifs et des SIP déployés sur ces environnements, ce qui offre aux concepteurs des SIP les moyens d'analyser de manière abstraite l'interaction entre un utilisateur et le système. Ce cadre conceptuel vise à aider le concepteur à identifier les descriptions des services que l'utilisateur a besoin dans son SIP.

Cet espace de services représente la partie *générique* de cette dimension conceptuelle puisqu'il représente un cadre conceptuel indépendant de la mise en œuvre des SIP et donc des modèles de contexte et des approches de services choisis pour son implémentation. Toutefois,

cette dimension conceptuelle intègre une partie spécifique nécessaire pour mettre en place notre vision intentionnelle et contextuelle des SIP. Nous proposons ainsi de décrire sémantiquement, selon cet espace de services, les services en incluant ces informations intentionnelles et contextuelles. L'objectif ici est de proposer des services de haut niveau, permettant de cacher la complexité technique de l'environnement due, entre autres, aux différentes technologies et services proposés. Ces services sont décrits selon le contexte dans lequel ils s'adaptent au mieux et les intentions qu'ils peuvent satisfaire. Ceci est basé sur une extension de OWL-S nommée OWL-SIC (*OWL-S Intentional & Contextual*) (*cf.* Chapitre 6). De plus, nous proposons dans ce cadre conceptuel de modéliser sémantiquement les informations contextuelles relatives aux éléments de l'espace de services et à l'utilisateur. Nous proposons une *ontologie multi-niveaux de contexte* (*cf.* section 6.4.1).

L'espace de services est présenté plus en détails, dans le cadre du Chapitre 5 qui illustre une conceptualisation de cet espace à travers la formalisation de ses différents éléments, services et capteurs de contexte, et la présentation de ses différentes caractéristiques, dynamisme et perméabilité.

4.3.2.2. Dimension fonctionnelle : Découverte et prédiction de services selon l'intention et le contexte

Un utilisateur interagit avec son SIP à travers l'espace de services. Il souhaite, dans le cadre d'un SIP, avoir des services qui répondent à ses besoins dans un contexte donné et en toute transparence. Ainsi, l'utilisateur expose son besoin au système, qui se charge de lui proposer des services en réponse à ce besoin et éventuellement à des besoins futurs (subséquents au besoin présent). Les services proposés sont sélectionnés comme étant ceux qui répondent au mieux à ses intentions dans son contexte courant et ceux considérés comme les plus appropriés en fonction de l'usage de l'utilisateur.

L'*intention* de l'utilisateur est valable dans un contexte donné tout comme le *contexte* influence la satisfaction de cette intention. Nous pensons, ainsi, que les deux approches, sensibles au contexte et intentionnelles, sont complémentaires et ne doivent pas être isolées les unes des autres. Nous pensons que, en tenant compte de ces deux aspects dans le cadre d'un SIP orienté services, il est possible de proposer à l'utilisateur des services qui peuvent mieux satisfaire ses intentions dans son contexte courant, renforçant ainsi la transparence du SIP. Cela peut être fait par la proposition de nouveaux mécanismes de découverte de services qui adoptent cette approche duale (intentionnelle et sensible au contexte), mais aussi par la proposition de nouveaux mécanismes de prédiction qui ont l'intention d'anticiper les besoins de l'utilisateur en fonction de ses intentions antérieures sur des contextes similaires.

Aujourd'hui, un enjeu important dans le domaine des SIP est de se positionner au niveau de l'utilisateur et de prendre en considération sa mobilité et le dynamisme de l'environnement qui l'entoure, sans pour autant perdre les moyens en termes de maîtrise et de contrôle. Ainsi, après une analyse de la littérature, nous avons constaté également que parmi les importants challenges des SIP, la *découverte et la prédiction dynamique des services* représentent deux

mécanismes permettant de satisfaire les besoins de l'utilisateur en lui offrant le service le plus approprié assurant ainsi la dynamique et la pro-activité des SIP. Toutefois, aucune des approches étudiées, par exemple (Toninelli et al., 2008)(Vanrompay et al., 2011)(Mirbel et Crescenzo, 2010a)(Olsson et al., 2011)(Santos et al., 2009)(Ramadour et Fakhri, 2011), n'exploite réellement l'étroite relation entre les approches intentionnelles et contextuelles. En conséquence, plusieurs possibilités sont offertes à l'utilisateur, qui n'est pas toujours en mesure de comprendre ce qui lui est proposé.

Selon notre étude de l'état de l'art, aucun des travaux analysés dans la section 3.5.2 ne propose un mécanisme de découverte de services qui combine et exploite réellement le contexte et l'intention. Un tel mécanisme de découverte de service guidé par le contexte et l'intention est nécessaire dans le cadre d'un SIP transparent et centrée utilisateur, et qui se caractérise par son adaptabilité au contexte et sa compréhension des besoins réels de l'utilisateur et de ses besoins. À notre avis, seulement un mécanisme de découverte de services basé à la fois sur le contexte et sur l'intention de l'utilisateur est en mesure de répondre à des questions telles que « *pourquoi un service est utile dans un contexte donné ?* » ou « *dans quelles circonstances émerge le besoin d'un service ?* ».

Une découverte dynamique des services est un mécanisme essentiel pour les SIP parce qu'il permet de répondre au mieux aux besoins de l'utilisateur. La prise en compte de la notion d'*intention* dans ce mécanisme permet de bien comprendre le réel besoin derrière la demande de l'utilisateur, appuyant ainsi le caractère centré utilisateur des SIP. De plus, la prise en compte de la notion de *contexte*, permet à ce mécanisme de découvrir le service qui pourra être réellement exécuté, lors de l'invocation du service, en se basant sur le contexte de l'utilisateur et sur le contexte dans lequel ce service est exécutable, assurant ainsi une meilleure adaptation à l'environnement. Ainsi la découverte de service joue un rôle important dans l'amélioration de la transparence des SIP, en proposant à l'utilisateur le service le plus approprié selon son intention et son contexte courant.

Toujours d'après notre analyse de la littérature (*cf.* section 3.5.3) les approches de prédiction se concentrent plus sur les mécanismes de prédiction de contexte suivant ou de prédiction de la continuité de contexte, sans pour autant souligner comment ce contexte prédit est utilisé pour découvrir et prédire le service suivant, par exemple. De plus, la plupart des systèmes de recommandation qui ont été étudiés se basent uniquement sur les informations contextuelles afin de proposer le service suivant aux utilisateurs, sans pour autant tenir compte des besoins réels de l'utilisateur derrière sa demande du service, c'est-à-dire, de ses objectifs. Ils proposent une certaine réalisation du service à l'utilisateur, tout en ignorant pourquoi ce service est nécessaire.

Nous soulevons ainsi le besoin de concevoir de nouveaux mécanismes de prédiction pour les SIP afin d'anticiper les besoins futurs de l'utilisateur en fonction de ses intentions antérieures sur des contextes similaires. Ce mécanisme introduit des techniques de recommandation afin d'augmenter le caractère dynamique et proactif des SIP en suggérant à l'utilisateur, d'une manière transparente, le service le plus approprié qui pourra, par la suite,

l'intéresser. L'objectif est de mieux comprendre les besoins de l'utilisateur afin de l'aider à y répondre d'une manière non intrusive. Ceci nous permettra d'offrir une meilleure pro-activité du système par la prise en compte de la relation entre la notion de contexte et les intentions, permettant ainsi de contribuer à l'amélioration de la transparence nécessaire des SIP.

Ainsi, afin de construire un SIP transparent, centré utilisateur et proactif, il est essentielle de se positionner au niveau de l'utilisateur afin de lui proposer les services les mieux adaptés à son contexte et qui répondent au mieux à ses besoins, ceci en se basant sur des mécanismes de découverte et de prédiction de services guidées par le contexte et l'intention.

4.3.2.3. Dimension système : Architecture de gestionnaire de SIP

Pour mettre en place notre vision intentionnelle et contextuelle des SIP conformément à l'espace de services, nous proposons une *architecture de gestionnaire de SIP* (*cf.* Chapitre 9) offrant essentiellement des modules de *gestion de contexte*, de *découverte de services* et de *prédiction de services* nécessaires pour l'utilisateur.

Cette architecture intègre nos différentes propositions pour la construction d'un SIP transparent et centré utilisateur :

- ***Un gestionnaire de contexte*** conforme à l'espace de services qui se chargera de l'acquisition des données contextuelles, à travers un ensemble de capteurs, et de la dérivation et de la modélisation de toutes les informations contextuelles des différents éléments de l'espace de services (services et capteurs) et celui de l'utilisateur (*cf.* section 9.3.2) ;
- ***La mise en œuvre du processus de découverte de services*** (*cf.* section 9.3.4), dont l'objectif est de satisfaire au mieux les besoins de l'utilisateur (formulés en termes d'intentions) dans son contexte courant, en lui proposant le service le plus approprié en toute transparence. Cette mise en œuvre se base sur l'implémentation d'un algorithme de mise en correspondance (*matching*) qui calcul le degré de mise en correspondance entre l'intention et le contexte courant de l'utilisateur et l'ensemble des services sémantiques décrit selon OWL-SIC (*cf.* Chapitre 6) et disponible dans le répertoire de services sémantique (*cf.* section 9.3.3) ;
- ***La mise en œuvre du processus de prédiction de services*** (*cf.* section 9.3.6), dont l'objectif est d'introduire des techniques de recommandation afin d'augmenter le caractère dynamique de l'architecture en suggérant à l'utilisateur, d'une manière transparente, le service le plus approprié qui pourra, par la suite, l'intéresser.

4.3.2.4. Dimension support : démarche méthodologique de conception des SIP

Pour la mise en œuvre et la réalisation de notre vision des SIP et afin de faciliter et d'organiser la conception de l'espace de services, nous proposons une démarche méthodologique de conception à destination des concepteurs des SIP. Cette démarche

supporte le passage du cadre conceptuel vers la description des services proposés au sein de l'architecture de gestionnaire de SIP. Notre objectif est d'aider la DSI à spécifier les fonctionnalités attendues de leur système, ainsi que les informations contextuelles qui seront capturées par celui-ci pour une meilleure adaptation. Il s'agit d'un processus de conception d'un espace de services et de ses différents éléments dans la perspective de garder le contrôle sur la définition du système et de ses services, tout en permettant la prise en compte d'un environnement hautement dynamique.

4.3.3. Contributions attendues

A travers ces différentes propositions, cette thèse vise à contribuer au domaine des SIP à différents niveaux :

- ***Au niveau conceptuel de notre vision d'un SIP :***
    - D'abord par la notion d'*espace de services* qui constitue un cadre conceptuel indépendant d'une approche spécifique de modélisation de contexte et de services, permettant la spécification des fonctionnalités proposées par un SIP à ses utilisateurs ;
    - La proposition d'un *cadre d'analyse et de comparaison des modèles de contexte*. L'objectif est d'aider à la compréhension et à l'analyse de la notion de contexte afin de mieux la représenter et la prendre en charge ;
    - L'extension de la *description sémantique des services OWL-SIC*, permettant la prise en considération de l'intention que le service permet d'atteindre et du contexte dans lequel ce service est valide et exécutable, ainsi que ces conditions contextuelles, dans les descriptions des services.
- ***Au niveau de l'implémentation de notre vision d'un SIP :***
    - La définition d'une *architecture de gestionnaire de SIP* orientée services, intention et contexte. Cette architecture intègre des modules de gestion de contexte, de découverte et de prédiction de services. Ces services, plus précisément leurs descriptions, sont enregistrés dans un répertoire sémantique de services, alors que les traces de leur usage sont enregistrées dans un répertoire de traces ;
        - La proposition et le développement d'un *processus de découverte de services*. Ce processus permet de découvrir et de sélectionner le service le mieux adapté afin de satisfaire le besoin de l'utilisateur dans un contexte donné, tout en masquant la complexité de l'environnement et en assurant une meilleure compréhension de l'utilisateur ;
        - La proposition et le développement d'un *processus de prédiction de services*, permettant de prédire, à partir de l'historique de l'utilisateur et

à partir de son intention et de son contexte courant, les services répondant à son intention future, avant même de le demander.

- ***Au niveau de la mise en œuvre de notre vision d'un SIP :***
    - La définition d'une *démarche méthodologique* permettant d'aider les concepteurs à définir leurs espaces de services, le modèle de contexte nécessaire, ainsi que la description sémantique de ces services ;
    - La proposition d'un *cas d'étude* permettant d'illustrer le déroulement des différentes étapes de ce travail de thèse et d'évaluer notre proposition.

4.4. CONCLUSION

Dans ce chapitre nous avons présenté un aperçu de la solution proposée dans cette thèse. L'objectif auquel doit répondre celle-ci est de concevoir et mettre en place un SIP transparent et centré utilisateur, permettant ainsi à un SI maitrisable d'évoluer dans le cadre d'un environnement pervasif.

Pour ce faire, nous avons d'abord introduit une nouvelle vision intentionnelle et contextuelle des SIP. Cette vision présente les SIP orientés services qui exploitent l'étroite relation entre l'intention et le contexte. Cette vision a été concrétisée, par la suite, selon quatre dimensions. La première dimension expose l'aspect conceptuel de notre solution. Nous proposons, dans cette partie, un cadre conceptuel permettant de guider la conception des SIP et de ses différents éléments. La deuxième dimension présente l'aspect fonctionnel de notre vision. Nous proposons dans ce cadre de nouveaux mécanismes de découverte et de prédiction de services répondants mieux aux attentes des utilisateurs selon leurs intentions et contexte. La troisième dimension expose l'aspect architectural de notre solution par la présentation d'une architecture de gestionnaires des SIP. Finalement, la quatrième dimension supporte le passage du cadre conceptuel à la mise en place de cette vision intentionnelle et contextuelle des services, par la proposition d'une démarche méthodologique.

Les chapitres suivants de cette thèse détaillent le contenu de cette solution et sont organisés de la façon suivante :

- Le *Chapitre 5* présente le cadre conceptuel à travers notre notion d'espace de services ;
- Le *Chapitre 6* explique notre description des services intentionnels et contextuels qui composent notre espace de services, en présentant notre extension de OWL-S ;
- Le *Chapitre 7* et *Chapitre 8* discutent et évaluent nos mécanismes de découverte et de prédiction de services selon l'intention et le contexte ;
- Le *Chapitre 9* décrit l'architecture de gestionnaire de SIP en présentant les composants essentiels qui la constituent et les interactions possibles entre ses composants ;
- Le *Chapitre 10* illustre notre démarche méthodologique de conception des SIP et présente un cas pratique d'application de notre approche.

Chapitre 5. Cadre conceptuel d'un SIP : Espace de Services

5.1. Introduction

Le but d'un Système d'Information Pervasif (SIP) est de rendre accessibles les fonctionnalités offertes par le SI à travers un environnement pervasif. Pour ce faire, les SIP doivent faire face à un certain nombre de problèmes, qu'on a cités dans le Chapitre 4, à savoir le *problème de transparence* lors de la gestion de l'hétérogénéité des environnements et des services, le *problème d'adaptation à l'environnement dynamique* tout en gardant la maitrise du SIP et le *problème d'adaptation à l'utilisateur* afin de répondre à ses besoins de la manière la plus appropriée et de manière complètement transparente pour l'utilisateur. Face à ces problèmes, la spécification d'un tel SIP devient ainsi un problème complexe à gérer. C'est d'autant plus vrai que l'immersion des fonctionnalités du SI dans un environnement pervasif expose celui-ci aux risques de perte de contrôle et de maitrise du SI dans un environnement pervasif fortement hétérogène et hautement dynamique. Appliquer un comportement totalement dynamique, avec (i) une prise en compte opportuniste des ressources disponibles dans l'environnement et (ii) l'autogestion des services offerts, peut constituer une menace pour le SI (par exemple, l'utilisation d'un composant non-certifié ou tout simplement, non-aligné aux stratégies de l'entreprise).

Avec le manque de modèle et de formalisme permettant de prendre en compte tous ces besoins de *transparence*, d'*adaptation à l'environnement* et d'*adaptation à l'utilisateur d'un SIP*, la DSI se trouve face à de grandes difficultés qui rendent difficile la conception et la réalisation d'un SIP transparent et centré utilisateur. Nous proposons pour lever ces difficultés un cadre conceptuel, que nous appelons *« espace de services »*, facilitant le passage d'un SI existant vers un SIP. Nous partons du principe qu'un SI traditionnel est déjà mis en place et que le futur SIP ne sera pas construit à partir de rien. Il s'agit surtout d'une évolution d'un SI vers un SIP en se basant sur un *espace de service*s. Celui-ci permet, d'une part, de masquer l'hétérogénéité caractérisant l'environnement pervasif et, d'autre part, de comprendre et de décrire les fonctionnalités proposées et leurs conditions d'usage indépendamment des technologies utilisées.

Ce cadre conceptuel, appelé *espace de services*, est construit selon le point de vue du SI en mettant en avant les composants majeurs de celui-ci qui sont (1) les *services* offerts à l'utilisateur et (2) les *capteurs* qui renseignent sur le contexte de l'utilisateur. Il est à noter que ces deux types de composants sont décrits d'une manière conceptuelle et indépendamment des aspects implémentations. De plus, ce cadre préconise aux fournisseurs de décrire les services qu'ils proposent en fonction des objectifs qu'ils permettent de satisfaire indépendamment de la manière dont ils sont réalisés.

Ce cadre conceptuel, appelé *espace de services*, représente un outil guidant la conception d'un SIP et l'identification des différents éléments qui le constituent, indépendamment de l'implémentation. Cette notion d'*espace de services* est définie dans l'optique de présenter un cadre conceptuel indépendant de la mise en œuvre des SIP et donc des modèles de contexte et des approches de services choisis pour son implémentation. Ce cadre conceptuel peut fonctionner sur n'importe quelle réalisation de SIP puisqu'il est générique et indépendant de la couche technique. Selon notre analyse de la littérature (*cf.* section 2.3.2), nous avons souligné quatre approches de modélisation de contexte, à savoir la modélisation basée sur les *clés-valeurs* (*cf.* section 2.3.2.1), la modélisation basée sur le *balisage* (*cf.* section 2.3.2.2), la modélisation orientée objet (*cf.* section 2.3.2.3) et la modélisation basée sur les ontologies (*cf.* section 2.3.2.4). De plus, nous avons détectés différentes approches de services, à savoir les approches des services Web (*cf.* section 3.4.1), des services sémantiques (*cf.* section 3.4.2) et des services intentionnels (*cf.* section 3.4.3). Ainsi, nous proposons un cadre conceptuel qui décrit d'une manière générique la notion de *contexte* et de *service* afin de fonctionner sur l'ensemble des modélisations de contexte et des approches de services citées ci-dessus.

Cet espace de services représente une vision conceptuelle de la dynamique et de l'évolution d'un SIP. Selon notre analyse de la littérature (*cf.* section 2.4), nous avons constaté que les SIP demeurent une nouvelle génération de SI sans un usage concret dans le milieu professionnel et qui jusqu'à présent n'ont pas été véritablement mis en place avec des formalismes appropriés. Ainsi, l'*espace de services* va permettre, d'une manière conceptuelle, de rendre explicite la dynamique de SIP, en représentant les différents éléments qui le composent à différents niveaux d'abstraction. Ceci va permettre d'avoir une idée sur le SIP à mettre en place ainsi que ses différents éléments, indépendamment de l'implémentation.

Dans le cadre de cette thèse, l'*espace de services* est un cadre conceptuel offrant un guide de conception pour identifier, décrire et valider les différents composants constituant l'espace de services en fonction des souhaits de l'entreprise et des utilisateurs ciblés par le SIP. Il représente un outil conceptuel pour aider le concepteur des SIP (la DSI), dont le but est d'assurer le maintien et le contrôle du Système d'Information même dans un environnement pervasif dynamique. Ce cadre de base est ensuite utilisé pour définir l'architecture de gestionnaire de services (*cf.* Chapitre 9) qui intègrera les mécanismes de découverte et de prédiction de services.

Cette notion d'espace de services nous permet d'analyser de manière abstraite l'interaction d'un utilisateur avec un SIP, comme l'illustre la Figure 25. L'utilisateur interagit avec le SI à travers l'espace de services dans lequel s'intègrent, de manière transparente, l'ensemble des services offerts par le SI et l'ensemble de capteurs qui renseignent l'utilisateur et le système sur le contexte.

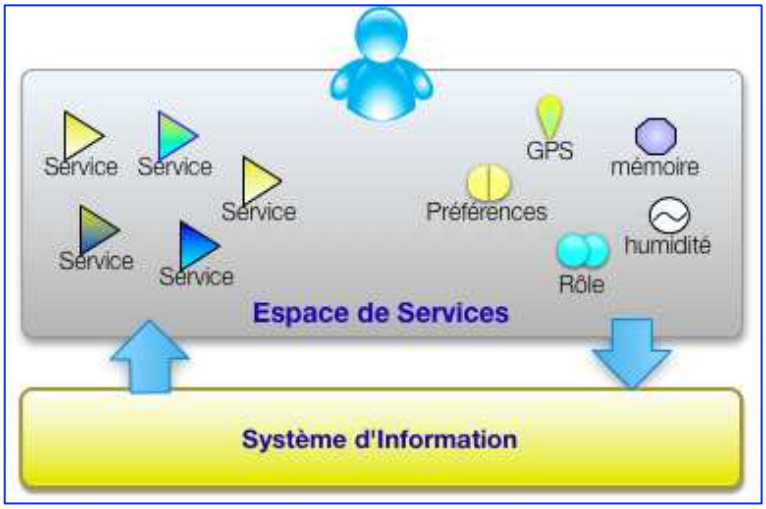


Figure 25. Dans un SIP, l'utilisateur interagit avec le système à travers un espace de services pervasif

Dans le cadre de ce chapitre, nous définissons la notion d'*espace de services* ainsi que les différents éléments qui le composent. L'ordre de présentation des éléments suit le point de vue du concepteur du SIP. Nous commençons par les fonctionnalités du système, qui seront représentées par le terme de **services**. L'environnement dans lequel évoluent ces services étant primordial dans un système pervasif, nous décrivons ensuite la manière dont nous modélisons cet environnement, sous le terme de **contexte**. Un service est choisi par rapport à son contexte et au contexte de l'utilisateur. Nous commençons par observer le contexte courant de l'utilisateur. Ce contexte correspond à l'observation d'un ensemble d'éléments qui entoure l'utilisateur à un instant t. Grâce à cette observation, nous sélectionnons par la suite le service qui répond au mieux à ce contexte de l'utilisateur. Le contexte courant de l'utilisateur est ainsi comparé aux conditions contextuelles associées aux services afin de sélectionner ceux qui peuvent opérer dans l'environnement courant de l'utilisateur. Ces informations relatives au contexte sont obtenues grâce à des **capteurs** sensibles à l'environnement. Ces capteurs font, selon nous, partie intégrante de l'espace de services et nous proposons une manière de les inclure dans cet espace au même titre que les services.

L'espace de services formalise le support contextuel que le SIP fournit à l'utilisateur. Ce support se concrétise à travers un ensemble de services offerts et un ensemble de capteurs permettant l'observation de contexte. Il est à noter que la formalisation de l'espace de services se focalise sur ces entités fournies par le SIP et l'utilisateur n'en est que le bénéficiaire.

5.2. Formalisation de la notion de service dans un SIP

La notion de service correspond à un concept largement répandu dans la littérature, défini de multiples façons. Parmi les nombreuses définitions existantes, et que nous avons présentées dans la section 3.2, nous citons, par exemple, la définition proposée par (Papazoglou et Georgakopoulos, 2003) (Papazoglou et al., 2008) qui considèrent le service comme un ensemble de modules logiciels autonomes apparu avec l'émergence du paradigme SOC comme une réponse au problème d'interopérabilité entre des applications et des

architectures hétérogènes. Une définition plus générale est donnée par (Issarny et al., 2007), qui considèrent un service comme une *entité indépendante, dotée d'interfaces bien définies et pouvant être invoquée de manière standard, sans requérir de son client une quelconque connaissance sur la manière dont le service réalise réellement ses tâches*.

Quelle que soit la définition utilisée, les aspects clés qui se dégagent des multiples définitions présentées dans la section 3.2 sont : le *faible couplage* entre le client et le fournisseur de service, l'*interopérabilité et la réutilisation*. En effet, ce faible couplage rend la notion de service particulièrement attractive pour les environnements pervasifs qui se caractérisent par la *volatilité* de leurs éléments (Vanrompay et al., 2011). De plus, la caractéristique principale des services est leur indépendance par rapport aux aspects technologiques, ce qui nous permet ainsi de masquer l'hétérogénéité technologique des environnements pervasifs. En effet, la notion de service a été conçue pour répondre à des besoins de réutilisation et d'interopérabilité (Papazoglou et Georgakopoulos, 2003). Ses récents usages dans les systèmes sensibles au contexte (Maamar et al., 2006)(Baldauf et al., 2007)(Toninelli et al., 2008) démontrent aussi l'intérêt de la notion de service pour l'adaptabilité dans les SIP.

Dans le cadre de notre *espace de services*, la notion de service représente une entité très importante. La notion de *service* est définie, d'une manière générale et dans toutes les approches orientées services, sous forme d'un *ensemble de fonctionnalités* $\mathcal{F}$. Comme nous l'avons montré dans le Chapitre 4, l'orientation intentionnelle et contextuelle permet de répondre aux problèmes de transparence, d'adaptation à l'environnement et d'adaptation à l'utilisateur auxquels les SIP sont confrontés. Ainsi, nous proposons dans ce cadre conceptuel des SIP de définir les notions d'intention et de contexte pour enrichir le concept de service.

5.2.1. Les fonctionnalités du service

Malgré les multiples définitions de la notion de *service* (*cf.* section 3.2) et les différentes approches de services proposées (*cf.* section 3.4), nous pouvons décrire, d'une manière générique, un *service* comme un ensemble de *fonctionnalités* $\mathcal{F}$ dont l'interface est clairement définie, voire standard, et dont le fonctionnement interne est inconnu des clients. La Définition 1 résume cette vision fonctionnelle d'un service.

Définition 1 :

Chaque fonctionnalité $f_j \in \mathcal{F}$ se définit en fonction des entrées in_j et des sorties out_j attendues par les clients du service.

$$\mathcal{F} = \{ f_j (in_j , out_j) \}$$

Définition 1. La notion de fonctionnalité relative à un service

D'un point de vue fonctionnel, chaque fonctionnalité f_j est définie en fonction de l'ensemble d'entrées *in* et de l'ensemble de sorties *out* attendues. La Définition 1 résume cet aspect de la notion de service. On remarquera qu'à aucun moment la nature exacte du service n'est révélée au client qui cherche et invoque un service sv_i. Seule l'interface fonctionnelle du service, représentée par l'ensemble de fonctionnalités f_j avec leurs données entrantes (*in*) et sortantes (*out*), est indiquée. Ce point est très important car il nous permet de gérer l'hétérogénéité de l'environnement, en gardant une certaine transparence dans la définition des services qui y sont offerts. Une découverte de services par les aspects fonctionnels est donc possible sans que le client ait besoin de connaître les technologies impliquées pour les réaliser. Par ailleurs, la nature atomique ou composite du service devient également transparente pour le client. En effet, les fonctionnalités exposées correspondent à celles offertes par le service. Un client peut alors les invoquer indistinctement, quelque soit leur origine, qu'elles viennent d'un service atomique ou d'une composition de services complexe.

Toutefois, afin d'améliorer la découverte de services et de proposer à l'utilisateur le service qui le satisfait au mieux, il est important de prendre en compte d'autres aspects, tels que les *intentions* que le service permet de satisfaire et le *contexte* dans lequel le service est valide et exécutable. Ces aspects jouent un rôle essentiel dans la découverte du service le plus approprié aux besoins de l'utilisateur. L'aspect intentionnel qui peut être associé à un service sv_i est introduit dans la section 5.2.2. Par contre, les aspects non-fonctionnels du service sv_i sont considérés sous l'angle contextuel, ci-dessous dans la section 5.2.3.

5.2.2. Les intentions du service

Les fonctionnalités proposées par un service permettent aux utilisateurs de répondre à un besoin précis. A partir de notre analyse de la littérature (*cf.* section 3.4.3.1), on peut associer ce besoin à l'expression d'une intention par les utilisateurs. Un service est ainsi proposé afin de satisfaire une (ou plusieurs) intention(s) de ces utilisateurs. La notion d'*intention,* présentée dans la section 3.4.3.1, formalise, en général, les besoins de l'utilisateur. Elle a été définie par Jackson (Jackson, 1995) comme étant une *déclaration 'optative' qui exprime ce que l'on veut, un état ou un résultat que l'on cherche à atteindre*. Ensuite, Kaabi et al. (Kaabi et Souveyet, 2007) présente l'intention comme *un but qu'on veut atteindre sans indiquer comment le faire*. Enfin, Bonino et al (Santos et al., 2009) l'a défini comme étant *un but à atteindre en effectuant un processus présenté comme une séquence de sous-buts et de stratégies vers le but cible*. En d'autres termes, une *intention* représente un besoin que l'utilisateur souhaite satisfaire, sans se soucier vraiment de quelle implémentation permet de le satisfaire ou de la façon de l'exécuter.

La notion d'intention a souvent été associée, par le passé, à la notion de service. Plusieurs travaux ont, en effet, considéré la notion de service sous un angle intentionnel (Santos et al., 2009) (Rolland et al., 2010) (Mirbel et Crescenzo, 2010a) et (Fensel et al., 2011), comme nous l'avons mentionné dans la section 3.4.3.1. Cette notion d'intention place donc le service à un niveau plus proche de l'utilisateur final : quelle que soit la technologie utilisée, le service est défini pour satisfaire un besoin exprimé par l'utilisateur sous forme d'intention, telle que

le présente la Définition 2. En tant que SI, les SIP doivent être conçus afin de mieux satisfaire les besoins des utilisateurs. Ainsi, les fonctionnalités d'un service doivent être soigneusement choisies afin de répondre aux besoins métiers exprimés sous forme d'intention.

Une intention peut être représentée selon une structure basée sur les travaux de (Prat, 1997), composée d'un *verbe*, une *cible* et un ensemble de *paramètres* relatifs au verbe (Rolland et al., 2010). Le verbe *v* caractérise l'action décrivant l'intention. Cette action agit sur une cible *tg*, pouvant être aussi bien l'objet affecté par l'action que le résultat obtenu par cette action. L'ensemble des paramètres *par* spécifient d'autres aspects de l'action tels que le bénéficiaire, la direction, la quantité, pour ne citer qu'eux. Ces paramètres sont, bien entendu, optionnels et dépendent directement du verbe utilisé. Par exemple, un service offrant à un agent commercial la possibilité de consulter une fiche client peut être associé à l'intention *I1 = {#consulter, #fiche_client, ∅}*. Ici, le verbe *« consulter »* est utilisé en compagnie de la cible *« fiche client »* sans qu'un paramètre soit indiqué. La formalisation de l'intention qu'un service permet de satisfaire est introduite dans la Définition 2.

Par exemple le service « *AccèsFicheClientServiceVPN* » est un service qui répond à l'intention principale I1 = *{#consulter, #fiche_client, ∅}*. Cette intention nécessite la réalisation d'autres intentions : I1.1 = *{#connecter, #VPN, ∅}*, I1.2 = *{#afficher, #liste_client, ∅}* et I1.3 = *{#sélectionner, #client, ∅}*. Ces intentions représentent les intentions qui vont être satisfaites par la mise en œuvre du service.

Chaque élément de cette définition doit être lui-même sémantiquement défini au préalable. En d'autres termes, nous supposons l'existence d'une ontologie d'intentions, qui décrit sémantiquement ces éléments. Cette ontologie, qui représente une ontologie du domaine, se compose, en réalité, de multiples ontologies, chacune décrivant un élément : une ontologie de verbes, une ontologie de cibles rendues accessibles par le SIP, ainsi qu'un ensemble d'ontologies spécifiant chacun des paramètres acceptés par le SIP. Ces ontologies établissent de manière non-ambiguë la sémantique des actions acceptées par le SIP dans l'espace de services et l'ensemble des cibles atteignables par le biais de cet espace.

Définition 2 :

Chaque intention I est définie par un verbe *v*, qui caractérise son action, une cible *tg*, sur laquelle l'action agit, et un ensemble optionnel de paramètres *par*.

$$I = \{ < v , tg , par > \}$$

Définition 2. Formalisation d'une intention qu'un service permet de satisfaire

Une telle définition, basée sur une ontologie préétablie, n'est envisageable que dans le cadre fermé d'un Système d'Information. En effet, ces systèmes n'autorisent pas un comportement ouvert sur des intentions et des cibles non-autorisées ou inconnues. D'autre

part, dans un environnement pervasif ouvert, l'expression de l'intention à l'aide d'un ensemble prédéfini de termes est difficilement imaginable, laissant une part trop importante à l'ambigüité. Cette ambigüité vient, en réalité, des utilisateurs : dans un environnement pervasif ouvert, le profil des utilisateurs n'est pas forcément connu à l'avance et le mode d'expression de ces utilisateurs peut être très différent.

5.2.3. Le contexte du service

Une intention n'est pas le fruit du hasard. Elle représente le besoin d'un utilisateur. Or ce besoin émerge dans un contexte donné. En d'autres termes, la notion d'intention est directement liée à la notion de contexte. Nous pensons qu'une intention n'a de sens que lorsqu'on la considère dans un contexte donné. Selon nous, et comme l'illustre la partie A de la Figure 26, un utilisateur invoque un service parce que celui-ci lui permettra de satisfaire une intention. Cependant, le contexte dans lequel émerge cette intention est lui-aussi significatif. Il peut influencer considérablement la manière dont cette intention peut être satisfaite, et donc influencer l'exécution même du service (par exemple, par le choix d'une implémentation qui s'adapte au contexte courant de l'utilisateur). Inversement, un utilisateur n'invoque pas un service uniquement parce qu'il est dans un contexte donné. Il le fait parce que ce service lui permettra d'atteindre ses objectifs dans ce contexte précis. La partie A Figure 26 schématise cette vision, selon laquelle un service est proposé afin de satisfaire une intention de l'utilisateur émergeant dans un contexte précis.

Cette vision commence à se développer dans la littérature, comme nous l'avons discuté dans la section 3.5.2.4. Quelques auteurs, dont Santos et al. (Santos et al., 2009), proposent déjà d'associer ces deux notions. Cependant, pour beaucoup d'entre eux, cette association reste assez floue. Par exemple, Santos et al. (Santos et al., 2009) ne considèrent le contexte que comme un filtre pour la découverte de services, le contexte étant vu comme une partie des entrées nécessaires aux services, et les intentions comme de simples étiquettes permettant de relier les demandes des utilisateurs aux services. Nous croyons, au contraire, que ces deux notions sont complémentaires et indissociables. Le contexte ne peut être réduit à de simples paramètres d'entrées ou de sorties. Non seulement il influence l'exécution du service, mais il caractérise aussi le service lui-même et les intentions ciblées par le service.

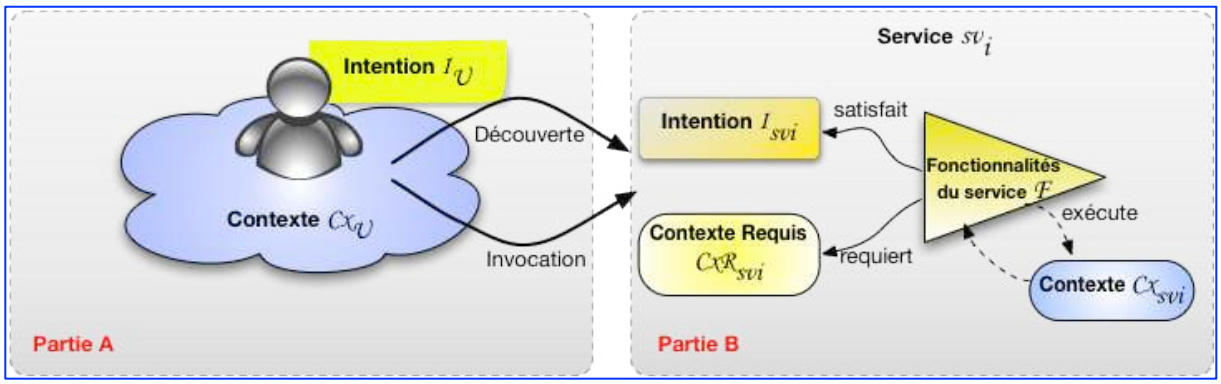


Figure 26. Relation entre l'intention, le contexte et les services.

A partir de la partie B de la Figure 26, on observe que l'impact de la notion de contexte sur le service est double : il influence son exécution et caractérise le service. Nous considérons qu'un service sv_i se trouve lui aussi immergé dans l'espace de services. Par conséquent, il se place lui-même dans un contexte Cx_{svi} donné. Ainsi, à l'instar de (Vanrompay et al., 2011), nous considérons qu'un service peut être associé à deux contextes complémentaires. Tout d'abord, un service s'exécute lui-même dans un contexte Cx_{svi} donné. Celui-ci représente un ensemble non vide d'observations contextuelles qui sert non seulement à indiquer les observations de contexte dans lesquelles le service est exécuté par son fournisseur, mais également à caractériser le positionnement de ce service dans l'espace de services. Le service est un objet dynamique qui bouge dans le temps et qui doit être observé. Ensuite, nous considérons également qu'un service sv_i peut avoir un contexte requis CXR_{svi} représentant un ensemble non vide de conditions contextuelles dans lesquelles le service est le plus apte à atteindre ses objectifs, *i.e.* les conditions de contexte auxquelles il peut s'adapter. En d'autres termes, le contexte requis CXR_{svi} représente un ensemble de conditions de contexte permettant au service une meilleure possibilité de satisfaction des intentions qui lui sont associées. Il s'agit d'un filtre défini sur les observations de contexte de l'utilisateur : plus le contexte courant observé pour l'utilisateur correspond au contexte requis, plus le service aura de chance de s'adapter à cette situation et de satisfaire au mieux l'utilisateur.

Ainsi, dans le cadre des mécanismes de découverte (*cf.* Chapitre 7) et de prédiction (cf. Chapitre 8) de services, le contexte requis est exploité dans le cadre de la sélection des services les plus appropriés au contexte courant de l'utilisateur (*cf.* section 7.2.2.4). Il faut mettre en correspondance le contexte requis du service (CXR_{svi}) avec le contexte courant l'utilisateur (Cx_U), afin de sélectionner les services qui peuvent s'adapter au mieux à la situation courante de l'utilisateur.

Dans le cadre de notre espace de services, la notion de contexte doit être modélisée d'une manière générique afin de permettre à n'importe quel modèle de contexte (*cf.* section 2.3.2) de fonctionner avec notre cadre conceptuel des SIP. Pour ce faire, nous avons besoin d'une modélisation de contexte de plus haut niveau qui peut être instanciée par n'importe quelle approche de contexte. Ainsi, afin de formaliser la notion de contexte nous proposons de définir avant un méta-modèle de contexte qui va être utilisé dans notre cadre conceptuel des SIP. Ce méta-modèle de contexte représente une modélisation générique de contexte Cx. Il permet à toute approche orientée contexte d'utiliser notre méta-modèle pour représenter ses informations contextuelles, quelque soit le modèle de contexte qu'elle va utiliser au niveau implémentation. Quant au contexte requis CXR_{svi}, il aura une structure similaire à Cx_{svi}, mais il doit exprimer des conditions contextuelles, lesquelles nécessitent un langage sémantique pour pouvoir les représenter sous la forme la plus adéquate. Ce langage sémantique va essentiellement dépendre du modèle de contexte choisi et mis en place au niveau implémentation. Ainsi, il est plus approprié de garder le choix de la modélisation de contexte CXR_{svi} au niveau implémentation et de spécifier ses conditions contextuelles en langage naturel au niveau conceptuel. En conséquence, au niveau conceptuel, il faut spécifier en langage naturel les conditions contextuelles qui vont être attachées à chaque service. Par la suite, en

choisissant le langage adéquat au niveau implémentation qui correspond au modèle de contexte choisi, il faudra traduire ces conditions pour pouvoir les traiter.

Nous présenterons le méta-modèle de contexte dans la section 5.3.1. Nous formalisons, dans la section 5.3.2, la notion de contexte à travers la notion d'observation et de capteur.

Ainsi, à partir de ce qu'on vient de présenter, il nous paraît évident qu'un service ne peut être défini sans tenir compte de ces éléments qui lui permettent de mieux se situer dans son environnement et de réagir à celui-ci. A ces éléments, on ajoute les notions préalablement mises en évidence : les intentions (Définition 2) et les fonctionnalités (Définition 1) offertes par le service. Nous formalisons ainsi la notion de service comme l'illustre la Définition 3

Définition 3 :

Un service sv_i correspond à un ensemble de fonctionnalités $\mathcal{F}$ fournies par cette entité sv_i dans un contexte $\mathcal{Cx}$ afin de satisfaire un ensemble d'intentions $\mathcal{I}$. La satisfaction de ces intentions dépend également d'un contexte favorable, décrit comme un contexte requis $\mathcal{CXR}$ pour le bon déroulement du service.

$$sv_i = <\mathcal{I}, \mathcal{F}, \mathcal{Cx}_{svi}, \mathcal{CXR}_{svi}>$$

Définition 3. Formalisation de la notion de service de l'espace de services

Derrière la notion de contexte, il existe une autre notion importante pour l'univers des services : les aspects non-fonctionnels qui caractérisent ces services. Beaucoup de recherches ont été menées autour de ces aspects (Liu et Issarny, 2004) (Chaari et al., 2008a) (Vanrompay, 2011)(Xin et Hao, 2012) (Ait Ali Slimane, 2012), pour ne citer qu'eux. Par exemple la notion de *qualité* présente l'un des aspects non-fonctionnels des services. Nous soulignons deux importantes tendances de travaux dans cette thématique, à savoir : la *qualité des service* (QoS – *Quality of Service*) et la *qualité de contexte* (QoC - *Quality of Context*). D'un côté, Vanrompay (Vanrompay, 2011), dans ces travaux de recherche, a souligné l'importance de la qualité de contexte (QoC) pour les applications, qui permet une utilisation efficace des informations contextuelles fournies. Cet auteur propose un ensemble de mesures de qualité des informations de contexte, telles que la probabilité d'exactitude, certitude, taux d'erreur standard, granularité, temps de réponse, etc. Plus spécifiquement, ces mesures représentent la qualité du futur contexte qui a été prédit (*QoFC – Quality of Future Context*) et elles sont classifiées en trois catégories de qualité de prédiction de contexte : *mesure individuelle*, *mesure globale* et *mesure dépendante des entrées*. De l'autre côté, Ait Ali Slimane (Ait Ali Slimane, 2012) a proposé une autre dimension qualitative des services sous un angle intentionnel, la *qualité des services intentionnels (QoiS – Quality of intentional Service)*. L'objectif principal derrière l'introduction de la QoiS est de résoudre les problèmes de discordance conceptuelle entre les exigences non-fonctionnelles des utilisateurs finaux et la qualité des services logiciels, celui des dépendances fonctionnelles entre les services. Dans

ces travaux, Ait Ali Slimane (Ait Ali Slimane, 2012) propose un méta-modèle de la qualité, et un *référentiel qualité*.

Cependant, il faut noter que le contexte est lui-même un de ces aspects. Le contexte peut être considéré comme un élément extérieur au service, capable de l'influencer. En tant que tel, il peut être considéré comme un ensemble d'aspects non-fonctionnels caractérisant le service. Inversement, beaucoup d'éléments traditionnellement pris en compte sous un angle d'analyse non-fonctionnelle peuvent également être vus comme des conditions contextuelles à l'exécution du service. A travers la spécification explicite du contexte offert $\mathcal{CX}_{svi}$ et requis $\mathcal{CXR}_{svi}$ par un service, nous espérons également offrir une possibilité de représentation des aspects non-fonctionnels dans le cadre de l'espace de services.

5.3. Formalisation de la notion de contexte

La notion de contexte, qu'on a discuté dans la section 2.3.1, est un concept très large, exploré depuis plusieurs années dans l'Informatique Pervasive (Masmoudi et Conan, 2013) (Najar et al., 2009) (Villalonga et al., 2010). Selon (Gensel et al., 2008), le contexte est « *l'ensemble des caractéristiques de l'environnement physique ou virtuel qui affecte le comportement d'une application et dont la représentation et l'acquisition sont essentielles à l'adaptation des informations et des services* ». Le contexte est un élément clé de l'Informatique Pervasive, car il est au centre des mécanismes d'adaptation prônés par les systèmes dits sensibles au contexte. Ces systèmes se caractérisent, en effet, par leur capacité à adapter leur fonctionnement afin d'augmenter leur utilisabilité et leur efficacité, par la prise en compte du contexte environnant (Baldauf et al., 2007).

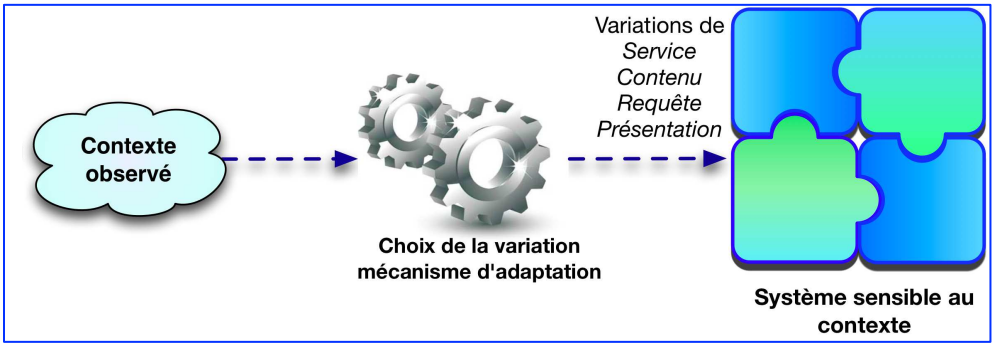


Figure 27. Vue schématique d'un système sensible au contexte (d'après (Najar et al., 2009))

De manière schématique, et comme l'illustre la Figure 27, un système sensible au contexte est un système qui supporte une certaine variabilité, le choix de la variante dépendant du contexte qui entoure l'exécution du système et son interaction avec les utilisateurs. Le contexte agit ainsi comme un élément extérieur au système qui influence sa variabilité intérieure, une sorte de paramètre qui guiderait le choix de la variante la plus appropriée et le processus d'adaptation la concernant (Najar et al., 2009).

Dans la section suivante, nous présentons une modélisation de contexte qui nous semble la plus générale permettant ainsi son exploitation et son extension dans différents domaines. Cette modélisation est le fruit de l'analyse des différentes approches de modélisation de contexte existantes, mentionnées à la section 2.3.2, et qui sont utilisées pour l'abstraction et l'encodage des informations de contexte dans une forme non ambiguë.

5.3.1. Modélisation de contexte

La notion de contexte (*cf.* section 2.3.1) est un élément essentiel pour la réalisation de systèmes sensibles au contexte. Or, tout Système d'Information Pervasif (SIP) doit également être sensible au contexte pour pouvoir s'adapter à l'environnement pervasif dans lequel évoluent ses utilisateurs. La maîtrise des SIP passe donc par la maîtrise de la notion de contexte. Celle-ci étant particulièrement extensible, une représentation formelle est nécessaire afin d'en définir les contours et d'identifier les informations pertinentes, qui influenceront effectivement le comportement du système (Gensel et al., 2008). Ce constat, effectué par différents auteurs (Gensel et al., 2008) (Masmoudi et Conan, 2013) (Chaari et al., 2007) (Brézillon et Brézillon, 2007) a conduit à la proposition de multiples modèles de contexte (Najar et al., 2009) (Bettini et al., 2010). Même si ceux-ci varient par leur forme et leur formalisme, nous pouvons dégager certains éléments clés, qui caractérisent l'immense majorité de ces modèles.

D'après l'analyse des différentes définitions et approches de modélisation de contexte, que nous avons exposées dans la section 2.3.2, nous remarquons certains éléments communs. En se basant sur la définition référence de Dey, « toute *information* qui peut être utilisée pour caractériser la situation d'une *entité* » (Dey, 2000), deux concepts clés peuvent être dégagés : (i) *« entité »* qui fait référence à quelque chose observable dans l'environnement et (ii) *« toute information »* qui est présentée comme tout attribut observé de cette entité. Ces concepts ont été soulignés dans la plupart des approches de modélisation de contexte, telles que (Lemlouma, 2004) (Kirsch-Pinheiro, 2006) (Paspallis, 2009) (Masmoudi et Conan, 2013).

En effet, dans la plupart des définitions de contexte, on parle de *l'observation de quelque chose afin de déterminer les informations de contexte*. Nous introduisons ainsi le concept *« sujet observé »* dans notre méta-modèle de contexte. Ce concept représente ce qu'on observe concrètement dans l'environnement afin de collecter un certain nombre d'informations contextuelles. C'est un élément important puisqu'il permet de déterminer à qui ou à quoi s'attache les informations de contexte que nous allons détecter. Lemlouma (Lemlouma, 2004), par exemple, dans sa modélisation de contexte se base sur l'observation de l'*utilisateur* et du *dispositif* afin de déterminer des informations sur leurs profils. L'utilisateur et le dispositif sont considérés comme des « sujets observés » dans sa modélisation de contexte CC/PP. Masmoudi et Conan (Masmoudi et Conan, 2013), ont souligné aussi l'importance de ce concept en introduisant dans leur méta-modèle la notion d'*entité observable* qui décrit une entité physique ou logique qui peut être observée.

De plus, la majorité des approches de modélisation de contexte expose et énumère dans leur modèle les éléments de contexte qu'ils souhaitent observer et acquérir pour pouvoir adapter leur système. Nous introduisons la notion *« élément de contexte »* qui représente l'attribut observé permettant de caractériser la situation du *« sujet observé »* en question. Kirsch-Pinheiro (Kirsch-Pinheiro, 2006), par exemple, propose une approche orientée objet pour la structuration des éléments de contexte et de leurs relations. Ce modèle propose une classe *élément de contexte* qui est spécialisée avec l'énumération des différents *éléments de contexte* qui sont liés aux aspects collaboratifs (rôle de l'utilisateur, activités, etc.) en plus des aspects physiques (la localisation de l'utilisateur, le dispositif, etc.). Dans cette modélisation l'élément de contexte est le concept central qui tourne autour des éléments qui semblent indispensables à l'auteur (Kirsch-Pinheiro, 2006), soit de par leur importance à l'utilisateur, soit de par leur pouvoir de caractériser une situation. De même, Reichle et al. (Reichle et al., 2008) soulignent aussi l'importance de ce concept *« élément de contexte »* dans la modélisation de contexte. Ils le présentent dans leurs travaux comme étant un *« scope »* identifiant l'attribut exact de l'entité sélectionnée qui le caractérise. Selon ce modèle de contexte, Reichle et al. (Reichle et al., 2008) associent directement leur notion du scope observé avec l'entité à laquelle il se réfère.

Dans un système sensible au contexte, l'observation d'un élément de contexte pour un sujet observé donné, permet de lui déterminer une valeur. Cette valeur est soit acquise dynamiquement à travers des capteurs, soit définie statiquement au départ ou dérivée ensuite à travers un processus d'interprétation. Compte tenu de l'importance de cette valeur dans le processus d'adaptation des systèmes sensibles au contexte, nous introduisons la notion de *« valeur observée »* dans notre méta-modèle qui est associée à un *« élément de contexte »*. Dans la littérature, par exemple, la modélisation de contexte constituée de paires *« clé-valeur »*, qu'on a présentée à la section 2.3.2.1, permet de représenter le contexte d'utilisation comme un ensemble de paires contenant chacune une clé et la valeur qui lui correspond. Dans cette approche, un élément de contexte est considéré comme *clé*, et la *valeur* associée à cet élément représente la valeur observée.

Finalement, nous avons constaté à travers la littérature que certaines informations supplémentaires sont associées aux informations de contexte modélisées pour les décrire. Paspallis (Paspallis, 2009), par exemple, souligne l'importance de la *représentation* des éléments de contexte lors de la modélisation. La *représentation* est utilisée pour préciser la structure interne utilisée pour encoder les informations de contexte dans des structures de données. Cet auteur propose ainsi de rajouter la *représentation* comme un concept clé de sa modélisation de contexte. De plus, Vanrompay (Vanrompay, 2011), souligne dans ces travaux l'importance de la prise en compte de la *qualité* de l'information de contexte utilisée. Cette *qualité* de contexte permet de déterminer la pertinence de l'information qui peut jouer sur le processus d'adaptation dans les systèmes sensibles au contexte. Dans cette optique, Pierrick et al. (Pierrick et al., 2013), ont proposé dans leurs travaux un méta-modèle de contexte qui souligne l'importance de la qualité de contexte lors de sa modélisation. Ils proposent le méta-modèle *QoCIM* qui offre un support de modélisation générique de critères de la qualité de contexte (QoC) en introduisant le concept *QoCIndicator*. Ainsi, une information de contexte

(*ContextData*) est qualifiée par des indicateurs de qualité (*QoCIndicator)*. Suite à ces propositions, nous soulignons de même l'importance de ces concepts, entre autres. Nous introduisons, en conséquence, le concept *« métadonnée »* comme étant les informations pertinentes qui peuvent être associées à la valeur observée d'un élément de contexte.

Nous présentons ainsi un méta-modèle flexible, illustré à la Figure 28. Ce méta-modèle permet de décrire comment les informations de contexte générales peuvent être modélisées. Il fournit une vue des structures de base qui peuvent être utilisées lors de la définition d'un modèle de contexte spécifique à un domaine donné. Il nous permet de « réduire » un modèle de contexte à l'observation d'un ou plusieurs *sujets observés* (un utilisateur, un dispositif, etc.) pour lequel (lesquels) on observe un ensemble d'*éléments de contexte* (localisation, activité, mémoire disponible, etc.), obtenant ainsi, pour chacun de ces concepts, des *valeurs* observées qualifiées par leur *métadonnée* (représentation, indicateurs de qualité, etc.).

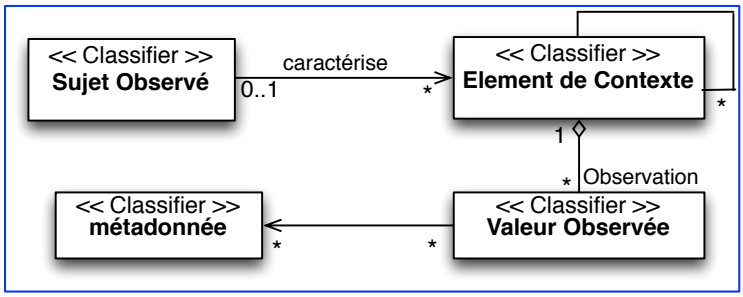


Figure 28. Méta-modèle de contexte.

Ce méta-modèle est fondé sur le principe de la séparation claire entre le *sujet observé* et *l'élément de contexte*. Le *sujet observé* représente le sujet de contexte auquel l'élément de contexte se réfère. L*'élément de contexte* fait référence à l'attribut ou à la propriété de contexte qu'on souhaite observer pour un sujet donné. Par exemple, on pourrait observer l'élément de contexte *« mémoire »* du sujet observé *« dispositif »*. Pour chaque sujet observé, on peut observer de zéro à plusieurs éléments de contexte. Un élément de contexte peut être associé à zéro ou plusieurs autres éléments de contexte. La représentation de cette *association* entre éléments de contexte, dans notre méta-modèle, permet d'exprimer les relations sémantiques qui peuvent exister entre ces éléments dans les approches *orientées objets*, *par balisage* ou *sous forme d'ontologie*. Cette relation entre éléments de contexte peut être une *composition*, une *spécialisation*, une *simple association*, etc. Ainsi, lors de l'instanciation de notre méta-modèle, il faut capturer la sémantique de ce lien qui peut exister entre les éléments de contexte dans le modèle de contexte au moment de son implémentation. Par exemple, dans une modélisation de contexte sous forme d'ontologie, on souhaite observer l'élément de contexte *« profil »* du sujet observé *« utilisateur »*. Cet élément de contexte se caractérise par d'autres éléments comme « *âge* », « *expérience* », etc. De ce fait, notre méta-modèle de contexte répond aux faiblesses des autres modèles de contexte à gérer l'aspect hiérarchique entre les éléments de contexte. Par exemple, (Kirsch-Pinheiro, 2006) propose un modèle de contexte très détaillé, où tous les éléments de contexte sont exposés directement dans le modèle. Ce modèle gère difficilement la hiérarchie entre les éléments de contexte. En

conséquence, des éléments de contexte comme « *LocalisationParGPS* » et « *LocalisationParAdresse* », par exemple, sont représentés comme deux éléments de contexte totalement distincts, alors qu'en réalité ils ne sont que des éléments composant de l'élément de contexte *« localisation »*. Ceci peut entrainer certaine confusion lors du traitement du modèle de contexte. Chaque élément de contexte est composé de plusieurs *valeurs observées*. Chaque valeur observée représente une donnée de contexte capturée à un instant donné. Par exemple, pour le *sujet observée* « *dispositif* » et l'*élément de contexte « mémoire »,* on pourrait avoir une *valeur observée* égale à *« 1024 Mo »*. Chaque valeur peut avoir des représentations différentes, des indicateurs de qualité, de confiance, etc. Ces derniers indiquent les aspects extra-fonctionnels des informations contextuelles représentées, dans le méta-modèle (Figure 28) comme des *métadonnées*. Ainsi une valeur observée peut avoir plusieurs métadonnées. Par exemple, on peut associer à une valeur donnée de la *localisationGPS* (élément de contexte) d'un utilisateur (*sujet observé*) la représentation *ReprésentationLocalisationGPS* et comme indicateur de qualité, la *confiance* à 80%.

Le méta-modèle que nous proposons dans le cadre de ce chapitre, peut s'appliquer aux quatre approches de modélisation de contexte citées dans la section 2.3.2. Par exemple, la modélisation de contexte basée sur les paires *clé-valeur* peut être instanciée par notre méta-modèle de contexte, où la *clé* est représentée par l'*élément de contexte* et la *valeur* est définie par la *valeur observée*. Si on prend, par exemple, la modélisation de contexte suivante : 'température' = '25'. Dans ce cas, la clé « température » représente l'élément de contexte auquel on associe une valeur égale à 25 qui peut être la valeur observée pour l'élément de contexte défini. Dans la modélisation *clé-valeur*, on n'a pas de relation entre éléments de contexte. De plus, les valeurs observées ne sont pas associées à des métadonnées. La cardinalité [0..n] entre éléments de contexte et entre une valeur observée et les métadonnées permet à notre méta-modèle de contexte d'être instancié par le modèle de contexte clé-valeur même s'il ne permet pas d'exprimer de relation sémantique entre les éléments de contexte et de lien entre les valeurs observées et les métadonnées.

Dans le cadre de la modélisation de contexte basée sur le balisage, nous prenons l'exemple de profil CC/PP présenté par Lemlouma (Lemlouma, 2004), et qui illustré à la Figure 29. Ce modèle de contexte représente une description des capacités physiques et logicielles d'un type de terminal donné. Ainsi, le *type du terminal* représente le *sujet observé* dans notre méta-modèle de contexte. Ensuite, pour chaque type de terminal une description de ces capacités lui est attribuée. Cette description porte sur un ensemble de composants (e.g *terminalHardware*, *hardwarePlatform*, *terminalSoftware*, *terminalBrowser*, etc.) auxquels est associé un ensemble de valeurs. Par exemple, pour le composant *hardwarePlatform* on lui associe comme nom *EPOC* et comme version *2.0*. A partir de cet exemple, nous pouvons affirmer que les *composants* représentés dans le modèle de contexte CC/PP représente les *éléments de contexte* de notre méta-modèle auxquels on va associer un ensemble de *valeurs observées*. De plus le lien hiérarchique entre éléments de contexte peut être représenté par le lien de *composition* entre composant dans le fichier CC/PP. Par exemple, le composant *terminalHardware* est composé du composant *hardwarePlatform.*

```
<ccpp:component>
    <rdf:Description rdf:about="http://www.example.com/profile#TerminalSoftware">
        <rdf:type rdf:resource="http://www.example.com/schema#SoftwarePlatform"/>
            <ex:name> EPOC </ex:name>
            <ex:version> 2.0 </ex:version>
            <ex:vendor> Synbian </ex:vendor>
    </rdf:Description>
</ccpp:component>
```

Figure 29. Extrait de profil CC/PP (Lemlouma, 2004)

Dans le cadre de la modélisation de contexte orientée objet, nous prenons l'exemple proposé par kirsch-Pinheiro (Kirsch-Pinheiro, 2006). Dans sa modélisation de contexte, cet auteur représente une classe *description de contexte* à laquelle on associe un ensemble de *valeurs* des *éléments de contexte* observés. Comme l'illustre la Figure 30, cette description de contexte décrit le contexte d'un élément de contexte donné (par exemple, le membre Alain). Ainsi, cette description de contexte peut être vue comme une *réunion d'un ensemble d'observation d'éléments de contexte* (*e.g.* session, rôle, application, etc.) *autour d'un sujet donné*. En conséquence, le lien entre un sujet observé et un ensemble d'éléments de contexte dans notre méta-modèle peut représenter cette description de contexte.

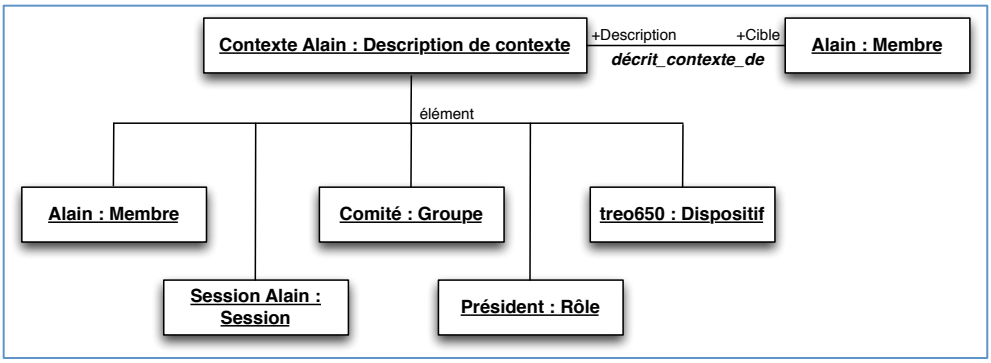


Figure 30. Extrait du modèle de contexte orienté objet de (Kirsch-Pinheiro, 2006) : lien entre sujet observé et élément de contexte

De plus, dans le modèle de contexte orienté objet de kirsch-Pinheiro (Kirsch-Pinheiro, 2006), et comme l'illustre la Figure 31, un élément de contexte peut avoir un lien avec un autre élément de contexte. Ce lien peut être un *« is-A »* ou une simple association entre deux éléments de contexte, par exemple l'élément de contexte « *session* » (ayant comme valeur *session Alain*) a un lien « *espace_d'éxécution* » avec l'élément de contexte « *disposition* » (ayant comme valeur *treo650*). Ceci représente le lien hiérarchique entre les éléments de contexte de notre méta-modèle de contexte. Dans cette modélisation de contexte, aucune métadonnée n'est prise en compte.

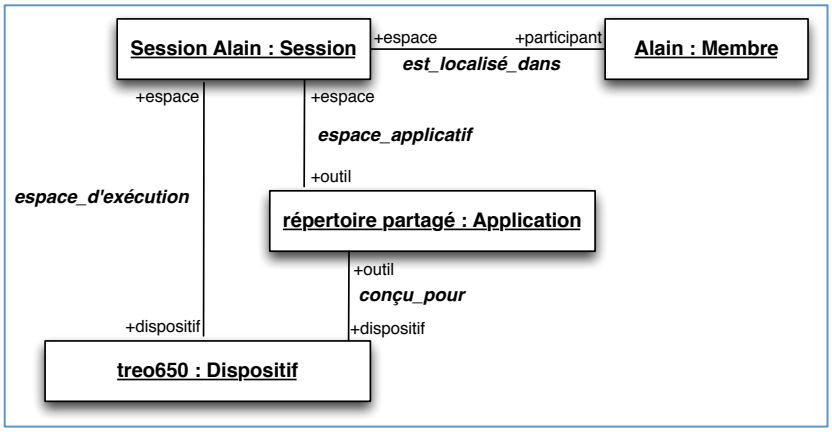


Figure 31. Extrait du modèle de contexte orienté objet de (Kirsch-Pinheiro, 2006) : lien entre éléments de contexte

Finalement, Reichle et al. (Reichle et al. 2008), par exemple, proposent une modélisation de contexte sous forme d'ontologie décrivant les informations contextuelles en se basant sur trois concepts principaux : 1) l'*entité* se référant à l'élément sur lequel l'information de contexte se rapporte ; 2) le *scope* identifiant l'attribut exact de l'entité sélectionnée qui le caractérise ; et 3) la *représentation* utilisée pour préciser la représentation interne utilisée pour encoder les informations de contexte dans des structures de données. En faisant le lien avec notre méta-modèle de contexte, l'*entité* peut représenter le sujet *observé*, le *scope* peut définir l'*élément de contexte,* la *valeur associée au scope* peut faire référence à la *valeur associée à l'élément de contexte*, et la *représentation* peut être une instance des *métadonnées.* De plus, la représentation du modèle de contexte sous forme d'ontologie permet de lier deux éléments de contexte ensemble. Ce lien peut être de type « *is-A* », « *Part-Of* » ou une relation de type *« property »* entre deux éléments de contexte. Ce lien représente une instance du lien hiérarchique entre éléments de contexte représenté dans notre méta-modèle de contexte

Dans la section, nous proposons une instanciation de notre méta-modèle de contexte en présentant notre modèle de contexte basé sur les ontologies.

5.3.2. Formalisation de la notion d'observation et de capteur

5.3.2.1. Formalisation de la notion d'observation

Selon notre analyse de la littérature (*cf.* section 2.3.2.4), nous avons conclu que les approches basées sur l'ontologie semblent être les plus prometteuses en termes de raisonnement et d'expressivité. Ainsi, le rattachement de notre méta-modèle de contexte à une description sémantique peut s'appliquer aussi bien aux éléments de contexte qu'aux sujets observés : avant toute observation d'un sujet ou d'un élément de contexte, la nature de celui-ci doit d'abord être clairement identifiée dans une ontologie de domaine propre au SI en question (*cf.* section 6.4.1). En d'autres termes, un SIP n'est pas sensé observer n'importe quel sujet, sans rapport avec son activité, de la même façon qu'il ne doit pas observer un élément de contexte inconnu, qu'il ne saura pas interpréter. Le fait de considérer uniquement des éléments spécifiés dans une ontologie présuppose que des éléments de contexte inconnus

jusqu'à alors ne seront pas pris en compte par le SIP, car ils risquent de ne pas être correctement interprétés par ce dernier. Ce n'est qu'après avoir été reconnu dans l'ontologie, et donc reconnu par le système lui-même, qu'un élément de contexte pourra être pris en compte. En effet, l'ontologie de contexte établit de manière non-ambiguë la sémantique des sujets et des éléments de contexte observables et acceptées par le SIP. Les SIP se situent avant tout dans le cadre fermé des Systèmes d'Information qui doivent rester maitrisés et contrôlés par la DSI. Par conséquence, ils n'autorisent pas un comportement ouvert sur des sujets et des éléments de contexte non-autorisés ou inconnus. Une fois cette description sémantique rendue possible par le modèle de contexte, nous pouvons nous concentrer sur l'observation de contexte par un capteur donné.

La notion d'*observation* est nécessaire parce qu'elle permet d'instancier notre méta-modèle de contexte au niveau instance avec des valeurs capturées à travers les capteurs et remontées ensuite au système. Nous pouvons ainsi dégager la Définition 4, qui établit la notion d'*observation* réalisée par un capteur. Pour un sujet donné *sj*, on peut observer un ensemble d'élément de contexte *Eso*. Une observation donnée s'attache à un seul élément de contexte *eo* observé pour le sujet *sj*. Ainsi, pour déterminer l'ensemble des éléments de contexte que nous pouvons observer pour un sujet *sj*, nous définissons la fonction suivante : *ElementsContexte* (*sj)* = *Eso*. La fonction *ElementsContexte* retourne pour un sujet donné *sj* l'ensemble de ces éléments de contexte observables. Au cœur de cette fonction, le type *so* du sujet observé *sj* est déterminé. De ce fait, l'association entre les éléments de contexte d'un sujet donné peut être formalisée selon cette fonction, mais sera différente selon les approches de contexte utilisées. Ainsi, cette relation entre les éléments de contexte sera plutôt exploitée au moment de la gestion de contexte.

Définition 4 :

Une observation O_{cpi} se réfère au capteur cp_i pour lequel on a observé, pour le sujet sj, un élément de contexte eo à un instant t_j. Ainsi, chaque observation est un n-uplet composé du sujet sj, de l'élément de contexte eo, ainsi que de la valeur *v* observée à un moment t_j et décrite par l'ensemble de métadonnées *M*.

$$O_{cpi} = \{ < ob_j , t_j > \},$$
$$ob_j = < sj , eo , v , M >, \text{ où}$$

- *sj* correspond au *sujet* observé ∈ *so* ;
- *eo* correspond à un *élément* de contexte ∈ *Eso* ;
- *v* correspond à une *valeur* observée pour ce concept ;
- *t* représente l'instant (*timestamp*) auquel cette observation a été réalisée ; et
- *M* correspond à l'ensemble des *métadonnées m* et de leur valeur *d* décrivant cette observation : M = {*m* = *d*}.

Définition 4. Formalisation de la notion d'observation

5.3.2.2. Formalisation de la notion de capteur

Un capteur offre au SI et aux utilisateurs un ensemble d'informations contextuelles correspondant à des valeurs observées dans l'environnement. Les capteurs, de différentes natures, permettent l'observation d'éléments caractérisant aussi bien l'environnement physique (GPS, température, etc.), que logique (mémoire disponible sur le terminal, préférences de l'utilisateur, etc.) et organisationnel (rôle de l'utilisateur, état d'exécution d'un processus, etc.). Ces capteurs sont importants puisqu'ils sont responsables de la capture des informations contextuelles dont le système aura besoin pour l'adaptation. Les *valeurs* observées correspondent à l'observation d'*éléments de contexte* clairement identifiés sur une ontologie de domaine (*cf.* section 6.4.1). Ces capteurs alimentent ainsi le SIP en informations contextuelles qu'il utilisera afin d'adapter son offre de services aux utilisateurs et à leurs besoins dans le contexte observé.

Un capteur cp_i est défini par l'ensemble d'observations O_{cpi} qu'il réalise, ainsi que par le contexte Cx_{cpi} dans lequel il se trouve. Ce contexte est lui aussi décrit par un ensemble d'observations d'éléments de contexte relatifs à un sujet donné, qui est, dans ce cas précis, le capteur lui-même. Le contexte de capteur Cx_{cpi} est un aspect important à prendre en considération car il permet de déterminer le positionnement du capteur dans l'espace de service. En effet, ce contexte Cx_{cpi} permet de déterminer à quel espace de services ces capteurs peuvent être attachés. La Définition 5 ci-dessous synthétise cette position.

Définition 5 :

Un capteur cp_i se définit en fonction d'un ensemble d'observations O_{cpi} qu'il réalise et d'un contexte Cx, également décrit par un ensemble d'observations.

$$cp_i = \{ O_{cpi}, Cx_{cpi} \}$$

Définition 5. Formalisation de la notion de capteur

5.4. FORMALISATION DE L'ESPACE DE SERVICES

Grâce à tous les éléments identifiés précédemment, nous proposons enfin de conceptualiser un *espace de services*, en définissant de manière générale chacun de ses éléments. L'espace de services est un cadre conceptuel qui permet de définir et de spécifier tous ce qui est jugé important et pertinent d'un point de vue utilisateur. Nous introduisons ainsi la notion d'*entité* comme une généralisation des différents éléments de l'espace : *services* (*cf.* section 5.2) et *capteurs* (*cf.* section 5.3). De plus, notre *espace de services* peut être étendu ou utilisé dans différents contextes. Cette généralisation va permettre ainsi de ne pas limiter cet espace aux notions de services et de capteurs mais de laisser la possibilité de prendre en compte d'autres éléments jugés pertinents dans d'autres usages.

Il est à noter que l'espace de services n'inclut pas l'utilisateur dans sa formalisation comme une entité mais les services et les capteurs de SIP doivent être définis pour supporter les intentions de l'utilisateur et capturer son contexte.

Plus spécifiquement, un espace de services ξ est un ensemble d'éléments, nommées *entités* (e_i), qui entourent l'utilisateur dans son environnement. Celui-ci est à la fois physique (localisation de l'utilisateur, dispositifs qui l'entourent, etc.), logique (ensemble des outils et des applications qui composent habituellement un SI) et organisationnel (organisation dans laquelle s'intègre le SIP). Dans cet environnement, nous pouvons considérer deux types d'entités distinctes : les *entités actives*, capables d'offrir aux utilisateurs un (ou plusieurs) service(s), et les *entités passives*, capables de renseigner les utilisateurs (et le système lui-même) sur cet environnement. En d'autres termes, nous proposons de généraliser la notion de *service* en la présentant comme une *entité active* dans un espace qui est capable d'agir sur l'environnement et la notion de *capteur* comme une *entité passive* permettant l'observation de cet environnement. A partir de cette considération, nous dégageons la Définition 6 suivante :

Définition 6 :

Un *Espace de Services* ξ est un environnement pervasif composé d'un ensemble d'entités e_i

$$\xi = \{e_i \mid e_i \in \mathcal{A} \ \vee \ e_i \in \mathcal{P}\}, \text{ où}$$

- $\mathcal{A}$ correspond à l'ensemble des *entités actives*, et
- $\mathcal{P}$ correspond à l'ensemble des *entités passives* disponibles sur l'espace ξ.

Définition 6. Formalisation de l'espace de service

Présentes dans l'espace de services, ces entités passives et actives se situent elles-mêmes dans un contexte donné, qui caractérise leur positionnement (au sens large du terme) dans l'espace de services ξ. Ainsi, et comme le synthétise la Définition 7, chaque entité possède un contexte Cx dans lequel on l'observe et qui lui est associé.

Définition 7 :

Toute entité e_i, qu'elle soit active ou passive, est associée à un contexte Cx qui la caractérise dans l'espace de services ξ.

Définition 7. la notion d'entité dans l'espace de services

De plus, grâce à la notion de contexte discutée précédemment et à la Définition 4, nous pouvons désormais définir le contexte Cx qui caractérise une entité (passive ou active) dans l'espace de services ξ. Nous considérons, dans la Définition 8, que les entités qui composent

l'espace de services ξ se trouvent elles-mêmes dans cet espace et peuvent donc être caractérisées par les informations contextuelles que l'on peut y capter à leur sujet.

Définition 8 :

Une entité e_i ($e_i \in \mathcal{A}$ ou $e_i \in \mathcal{P}$) est caractérisée, dans l'espace de services ξ, par un contexte Cx. Celui-ci est constitué d'un ensemble d'observations. Chaque observation se réfère à l'entité e_i observée et contient une valeur v pour un élément de contexte eo observé à un instant t, ainsi que l'ensemble de métadonnées $\mathcal{M}$ caractérisant cette observation.

$$Cx = \{ < ob_j, t_j > \},$$
$$ob_j = < e_i, eo, v, \mathcal{M} >$$

Définition 8. Le contexte qui caractérise une entité dans l'espace de services

5.4.1. Entités actives

Le concept de *service* permet d'évoquer le caractère *actif* des entités capables d'offrir à l'utilisateur et aux autres entités placées dans l'espace de services un ensemble de fonctionnalités agissant sur l'environnement qui les entoure. Ainsi, nous pouvons généraliser la notion de *service* en une *entité active*. A travers la notion de service, la nature réelle des *entités actives* demeure cachée derrière une interface bien définie, représentant les fonctionnalités pouvant être rendues par ces entités, sans spécifier comment elles le sont. En définissant les entités actives en tant que services, nous pouvons gérer l'hétérogénéité qui les caractérise dans l'espace de services.

L'ensemble des entités actives $e_i \in \mathcal{A}$ correspond à l'ensemble des *services* offerts aux utilisateurs par le SI dans un environnement pervasif. Une entité active est une entité capable de réaliser une action (atomique ou composée de plusieurs actions) pour un client quelconque à travers une interface clairement définie. Le client n'a pas besoin de connaître les détails de l'implémentation du service, mais uniquement son interface. Les entités actives sont en mesure d'agir sur l'environnement, qu'il soit physique (gestion de la température d'un dépôt de stockage, par exemple), logique (invocation d'un service Web ou d'une fonctionnalité d'un ERP) ou organisationnel (*e.g.* intervention sur un processus métier). La nature exacte de l'entité active est ainsi transparente : qu'il s'agisse d'un service Web, une imprimante ou un fax, l'entité sera toujours vue à travers les services qu'elle offre à ces clients.

5.4.2. Entités passives

Le concept de *capteur* permet d'évoquer le caractère *passif* des entités capables d'observer l'environnement et de renseigner l'utilisateur et le SI sur les informations contextuelles observées de cet environnement. Ainsi, nous pouvons généraliser la notion de *capteur* en une

entité passive. Ces entités passives sont importantes puisqu'elles sont responsables de la détection des informations contextuelles dont le système aura besoin pour l'adaptation. Leur définition délimite la notion de contexte, spécifiant les informations considérées comme pertinentes. L'objectif est de permettre la prise en compte de ce contexte afin de proposer aux utilisateurs les services qui leur correspondent le mieux.

L'ensemble des entités passives $e_i \in \mathcal{P}$ correspond à un ensemble de *capteurs* capables d'observer l'environnement de l'utilisateur et son interaction avec le SIP. En d'autres termes, une *entité passive* est une entité capable d'offrir au SI et aux utilisateurs un ensemble d'informations contextuelles correspondant à des valeurs observées dans l'environnement, qu'il soit physique, logique ou organisationnel. Il s'agit d'une source d'information qui alimente les autres éléments de l'espace de services en informations contextuelles. Ces informations peuvent, par la suite, être utilisées pour l'adaptation des services offerts à l'utilisateur, voire l'adaptation de l'espace de services lui-même. En d'autres termes, ces entités passives alimentent le SIP en informations contextuelles qu'il utilisera afin de mieux adapter son offre de services aux utilisateurs et à leurs besoins dans le contexte observé.

5.4.3. Etat de l'espace de services et son évolution

L'espace de services représente un espace conceptuel et non technique constitué d'un ensemble d'entités (services et capteurs) à un instant *t*. En réalité, cet espace de services ne forme pas un espace discret et fermé. Bien au contraire, nous proposons un espace de services *dynamique* qui n'est pas formalisé ou défini par une description propre à lui mais par la description des différentes entités qui le constituent à un instant donné. En conséquence, un espace de services peut *évoluer* dans le temps, avec de nouvelles entités qui s'ajoutent aux entités déjà présentes et d'autres entités qui disparaissent ou deviennent simplement indisponibles. On peut ainsi distinguer deux visions à partir de cette définition. D'abord, une *vision statique* qui conceptualise l'espace de services par la définition des entités dont l'intégration à l'espace est prévue, voire souhaitée par l'entreprise. Or, la nature fortement dynamique des environnements pervasifs oblige cette conceptualisation à évoluer avec cet environnement. Une *vision dynamique* est donc nécessaire. Celle-ci doit tenir compte des entités effectivement disponibles dans l'espace de services à un instant t. Cette distinction est nécessaire afin de mieux capturer la dynamique propre à cet espace. La notion d'espace de services se présente ainsi comme un *conteneur conceptuel* permettant aux concepteurs de SIP de mieux décrire ces systèmes. Dans les faits, les SIP sont confrontés à la dynamique propre aux environnements pervasifs, caractérisée par la volatilité des ressources qui s'y trouvent. Cependant, contrairement aux environnements purement pervasifs, pour lesquels la découverte et la prise en compte opportuniste des entités est recherchée, les SIP requièrent le maintien d'un certain contrôle et maîtrise. Il s'agit, avant tout, d'un système d'information, et la prise en compte de nouvelles entités sans une validation ou un accord au préalable de la part de l'entreprise (typiquement, de sa DSI), n'est pas toujours possible, voire n'est pas souhaitable. La notion d'espace de services permet aux concepteurs de mieux imaginer les environnements optimaux, maîtrisés et pourtant dynamiques. Les concepteurs peuvent ainsi décrire leur SIP sous la forme de multiples espaces de services perméables, comme l'illustre

la Figure 32. En d'autres termes, un espace de services n'est pas un espace fermé totalement déconnecté des autres espaces. Bien au contraire, c'est un espace qui n'a pas de frontières claires qui l'empêchent de communiquer avec les autres espaces et qui demeure accessible. Ainsi, ces espaces de services peuvent se partager des entités (actives ou passives) communes (voir Figure 32). Les entités actives ou passives d'un espace peuvent ainsi exister sur d'autres espaces. De plus, l'utilisateur évolue entre ces multiples espaces qui se superposent et évoluent dans le temps.

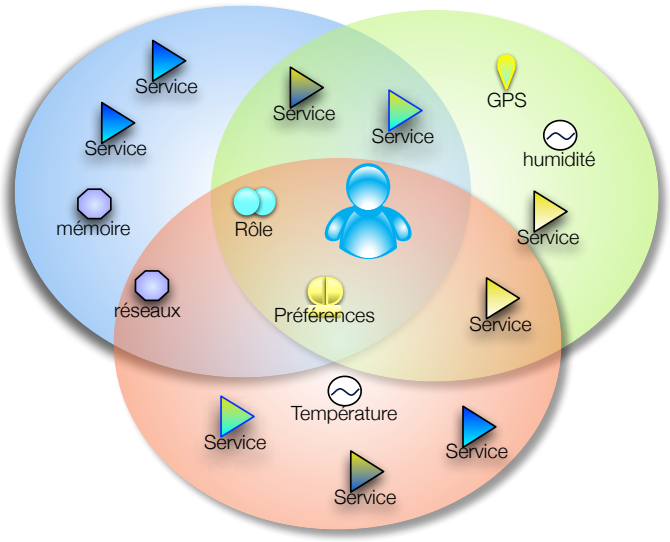


Figure 32. Multiples espaces de services perméables

Afin de permettre aux visions statiques et dynamiques de coexister de manière harmonieuse, nous considérons l'état d'un espace de services en plus de sa définition statique, selon la Définition 9. L'*état* d'un espace de services ξ à un instant *t*, noté ξ^t, correspond aux entités, passives ou actives, effectivement disponibles sur l'espace ξ à cet instant. Ceci équivaut à dire qu'une entité e_i possède elle aussi un état à un instant *t*. Cet état de l'entité, noté $e_i^{\xi t}$, indique la disponibilité de celle-ci dans l'espace ξ à l'instant *t*.

Définition 9 :

L'*état* d'un espace de services ξ à un instant *t*, noté ξ^t, se définit comme l'ensemble des états des entités e_i présentes à cet espace :

$$\boldsymbol{\xi^t \subseteq \xi,\ \xi^t = \{e_i | e_i \in \xi \wedge e_i^{\xi t}\}}, \text{ où}$$

$e_i^{\xi t}$ indique l'état de l'entité e_i à l'instant *t* (disponible ou indisponible).

Définition 9. Etat d'un espace de services

5.5. CONCLUSION

Dans ce cinquième chapitre, nous avons proposé la notion d'*espace de services*, cadre conceptuel pour la conception d'un SIP. Celui-ci permet la définition d'un SIP par la spécification d'un ensemble d'espaces, dans lesquels cohabitent les services offerts par le système aux utilisateurs et les capteurs qui l'alimentent avec des informations récoltées à partir de l'environnement (physique, logique ou organisationnel). Ces éléments sont définis par rapport à un contexte dans lequel ils se situent, permettant ainsi une meilleure prise en compte de l'environnement dans lequel évolue l'utilisateur. Cette notion, représentant notre vision des SIP, permet aux DSI de spécifier les fonctionnalités attendues de leur système, ainsi que les informations qui seront capturées par celui-ci pour une meilleure adaptation. Il s'agit de garder le contrôle sur la définition du système et de ses services, tout en permettant la prise en compte d'un environnement hautement dynamique. Ce compromis entre dynamisme et spécification au préalable est une nécessité pour les SIP, car, même s'ils veulent tirer profit d'un environnement pervasif, ils restent néanmoins des SI. Cette possibilité de spécifier le système tout en permettant l'expression de sa dynamique constitue une des forces de cette proposition.

L'*espace de services* est défini comme un cadre conceptuel, lequel formalise d'une manière *générique* la notion de *contexte* et de *service*. Ce cadre est conçu *indépendamment de la couche technique*, ainsi que des approches de modélisation de contexte et des approches de services choisies pour son implémentation. En réalité, l'objectif principal de cet espace de services est de représenter une *vision conceptuelle de la dynamique d'un SIP*, offrant ainsi un guide de conception pour *identifier, décrire et valider les différents composants* constituant cet espace en fonction des souhaits de l'entreprise et des utilisateurs ciblés par le SIP.

Nous avons formalisé cette notion d'espace de services et présenté ses composants dans l'ordre qui nous semble le plus pertinent du point de vue d'un DSI. Le cadre conceptuel proposé dans ce chapitre s'inscrit dans une approche de recherche plus large. D'une part, cette approche inclut la mise en œuvre de cette vision des SIP à travers la *description sémantique des services,* intégrant les notions d'intention et de contexte (*cf.* Chapitre 6), *la découverte* (*cf.* Chapitre 7) et *la prédiction de services* (*cf.* Chapitre 8) basées sur les notions de l'espace de services. Cette mise en œuvre a été mise en place par la réalisation d'une architecture que nous avons développée dans le cadre de cette thèse pour définir les différents espaces de services et leurs entités, et que nous avons nommée architecture de gestionnaire d'un SIP (*cf.* Chapitre 9). D'autre part, cette approche inclut une démarche de mise en œuvre et de réalisation de cet espace à l'intention des concepteurs de SIP (*cf.* Chapitre 8). Cette démarche supporte le passage du cadre conceptuel « espace de services » vers l'implémentation de l'architecture de gestionnaire de SIP contenant les modules de description, découverte et prédiction de services, que nous proposons dans la suite de cette thèse.

Chapitre 6. DESCRIPTION INTENTIONNELLE ET CONTEXTUELLE DES SERVICES

6.1. INTRODUCTION

La *vision intentionnelle et contextuelle des SIP* que nous proposons dans le cadre de ce travail de thèse (*cf.* section 4.3.1) met l'accent sur l'étroite relation entre l'*intention*, le *contexte* et le *service*. Nous considérons qu'un service est là pour répondre à une intention particulière dans un contexte donné. Ce contexte peut influencer la satisfaction de cette intention. Cette vision a été mise en place conceptuellement par la notion d'*espace de services*, que nous avons présentée dans le chapitre précédent (*cf.* Chapitre 5).

L'espace de services présente un cadre conceptuel qui intègre les différents services et capteurs qui entourent l'utilisateur dans son environnement. Nous décrivons cet espace par une description formelle qui lui est propre laquelle considère l'ensemble de services potentiellement disponibles à un instant donné. Ceci est dans le but de gérer l'aspect dynamique et perméable de cet espace.

Une description de services selon une vision intentionnelle et contextuelle est un élément clé pour la réalisation de cette vision. Un service doit être décrit selon l'intention qu'il est capable de satisfaire. Il doit être décrit également par le contexte dans lequel il est valide et exécutable. Une telle description reflète l'étroite relation entre la notion d'intention, celle de contexte et le service. Son rôle est d'apporter plus de signification au service (intention) et de prendre en considération son adaptation à l'environnement (contexte). Ce descripteur de services représente une extension de OWL-S, que nous avons appelé OWL-SIC *(OWL-S Intentional & Contextual)*, laquelle considère le service comme une entité unique avec de multiples dimensions : dimensions intentionnelles, techniques et contextuelles.

Le descripteur intentionnel et contextuel que nous allons décrire dans ce sixième chapitre permet la description de l'espace de services à travers la description de l'ensemble des services qu'il intègre. De plus, ce descripteur représente le point d'entrée pour le mécanisme de découverte (*cf.* Chapitre 7) et celui de prédiction (*cf.* Chapitre 8) de services. Ces mécanismes interagissent avec un répertoire de services contenant ces descripteurs pour satisfaire les besoins des utilisateurs.

Dans le cadre de ce chapitre, nous argumentons notre choix d'étendre le langage OWL-S pour prendre en considération les notions de *contexte* et d'*intention*. Par la suite, nous détaillons notre descripteur de services et nos extensions de OWL-S.

6.2. VERS UN DESCRIPTEUR INTENTIONNEL ET CONTEXTUEL : OWL-SIC

Selon notre vision intentionnelle et contextuelle des Systèmes d'Information Pervasifs (*cf.* section 4.3.1), les services sont sélectionnés et exécutés dans un contexte donné et sont censés satisfaire certaines intentions particulières. Ces services forment les entités actives de notre *espace de services* (*cf.* section 5.4.1). Cet espace de services représente un cadre conceptuel, non technique, qui se définit, entre autres, à travers les différents services qu'il englobe. Ainsi, il est important de présenter une description des services selon cette perspective intentionnelle et contextuelle.

Les descriptions de services actuelles ne décrivent pas nécessairement les deux aspects. Pour beaucoup de travaux, cette association reste floue et pas assez exploitée. Ce constat se base sur l'analyse des approches de découvertes de services détaillée dans la section 3.5.2.

Ainsi, nous proposons un descripteur intentionnel et contextuel de services qui supporte notre notion d'*espace de services*. Ce descripteur correspond à la définition et à la description des différentes entités actives (services) qui cohabitent dans un ou plusieurs espaces de services. Pour cela, nous proposons de relever les services à un niveau plus élevé en proposant une description sémantique qui inclut la description intentionnelle et contextuelle des services. Un utilisateur n'a pas besoin d'un service parce qu'il est situé dans un endroit donné ou dans un contexte donné. Il nécessite un service parce qu'il a une intention particulière qu'un service peut satisfaire dans ce contexte. Cette description sémantique étendue des services, permet de gérer l'hétérogénéité de l'environnement. L'objectif ici est de proposer des services de haut niveau, permettant de cacher la complexité technique de l'environnement dû, entre autres, aux différentes technologies et services proposés.

Afin de proposer une telle description sémantique, nous avons besoin d'un langage assez riche et suffisamment extensible pour prendre en considération les aspects intentionnels et contextuels dans la description d'un service. Dans le Chapitre 3, nous avons évoqué deux langages de description de services correspondants à ces critères. Le premier est le langage WSMO (*cf.* section 3.4.3.2) qui représente une ontologie décrivant sémantiquement les différents aspects relatifs à un service Web. WSMO (Keller et al., 2004) se caractérise essentiellement par son approche basée sur les intentions. Celle-ci prend en considération les intentions spécifiques derrière une recherche de services de l'utilisateur. Le deuxième langage est le langage OWL-S (Martin et al., 2007). Celui-ci (*cf.* section 3.4.2.1) définit les *capacités* des services Web en trois parties représentées par des ontologies interdépendantes : le *profil de services*, le *modèle de processus* et le *grounding*.

Face à ces deux propositions, nous avons considéré la question suivante : quel langage parmi les deux cités ci-dessus (OWL-S et WSMO) permettrait *de décrire, d'une façon claire, expressive et sans perte d'informations, l'intention que le service est capable de satisfaire et le contexte de validité et d'exécution de ce service, et qui permet également de raisonner dessus*. Selon Roman et al. (Roman et al., 2005), dans WSMO, une intention décrit les aspects

liés aux désirs de l'utilisateur par rapport à la fonctionnalité demandée. Toutefois, cette intention n'est pas formulée selon un modèle bien spécifique. Comme nous l'avons mentionné dans la section 3.4.3.2, cette information est représentée uniquement comme un ensemble d'objets. Par conséquent, WSMO ne permet pas d'identifier le rôle réel que joue chaque objet dans la spécification de l'intention. En d'autres termes, ce langage n'exploite pas la sémantique des verbes, des cibles et des paramètres qui peuvent représenter une intention, comme présenté dans les travaux de (Kaabi et Souveyet, 2007) et (Rolland et al., 2010). Nous concluons ainsi, que cette représentation d'intention perd une partie de son expressivité avec une telle modélisation. Dans notre vision, l'intention représente un élément clé très riche et qui apporte une réelle compréhension au service associé. Il est important de décrire la sémantique de tous éléments composant cette intention pour tout éventuel usage, tel que la découverte et la prédiction de service. De plus, une des reproches de WSMO est qu'il ne s'appuie pas sur des moteurs de raisonnement qui soient assez performants et puissants pour raisonner sur l'intention et le contexte. Cette phase de raisonnement est très importante dans nos mécanismes de découvertes (*cf.* Chapitre 7) et de prédictions de services (*cf.* Chapitre 8), étant donné que nous raisonnons sur les ontologies et sur les descriptions de services.

Concernant le langage OWL-S (*cf.* section 3.4.2.1), les dernières recherches dans l'Informatique orientée services recommandent l'utilisation de ce langage pour décrire sémantiquement les services. Selon (Suraci et al., 2007), même si OWL-S est conçu pour les services Web, il est riche et suffisamment général pour décrire tout type de services. Le langage OWL-S représente un langage souple et facilement extensible, tel que démontré par les travaux de Suraci et al. (Suraci et al., 2007) et Vanrompay et al. (Vanrompay et al., 2011). De plus, OWL-S offrent un très grand pouvoir d'expressivité ainsi que la possibilité d'inférer de nouvelles connaissances grâce au nombre de constructeurs de ce langage ainsi qu'au nombre de raisonneurs proposés, tels que *Jena* (Carroll et al., 2004), *Pellet* (Parsia et Sirin, 2004) et Racer (Haarslev et Möller, 2003).

Ainsi, nous avons choisi de nous baser sur OWL-S puisque, d'une part, il est assez souple et extensible pour inclure d'autres types d'informations dans la description de services, et d'autre part, parce qu'il offre des moteurs de raisonnement assez puissants nécessaires pour la découverte et la prédiction de ces services. Nous proposons ainsi d'étendre OWL-S afin d'inclure les informations concernant à la fois le contexte et l'intention qui caractérisent un service. Cette extension est présentée sous le nom de OWL-SIC (*Intentional & Context aware Services Web Ontology Language)*. Elle a fait l'objet de plusieurs publications (Najar et al., 2011a) (Najar et al., 2011b) (Najar et al., 2012c) et sera utilisée, dans la suite de cette thèse, pour la découverte (*cf.* Chapitre 7) et prédiction de services (*cf.* Chapitre 8).

Pour inclure l'aspect intentionnel à la description de services, nous avons différentes manières de faire et différentes extensions possibles. La première extension, et la plus simple, consiste à *étendre la description du profil de services*. La problématique ici est que cette description intentionnelle peut rapidement devenir complexe si on prend en considération la description de la variabilité dans l'expression de l'intention (*cf.* section 6.3). Ainsi, par soucis d'interopérabilité, nous avons choisi d'apporter des modifications minimes à la partie du

profil de service, laquelle est utilisée par les mécanismes de découverte traditionnels, afin de garder son caractère souple et flexible. Nous rajoutons ainsi un nouveau bloc « *Intention service »* dans la description de services, lequel se concentre uniquement sur la description intentionnelle des services. Par ailleurs, et à l'instar de Kirsch-Pinheiro et al. (Kirsch-Pinheiro et al., 2008), nous avons choisi de décrire les informations relatives au contexte par une URL qui fait référence à une ressource externe, permettant ainsi une mise à jour facile des informations de contexte liées à la description du service (*cf.* section 6.4).

Avec notre extension OWL-SIC, nous pouvons décrire l'intention qu'un service est capable de satisfaire et les conditions de contexte dans lesquelles ce service est valable et peut être exécuté. Par ailleurs, contrairement à Rolland et al. (Rolland et al., 2010), nous ne considérons pas que le service intentionnel doive être observé comme une entité distincte du service technique. Cette séparation conduit à des descriptions techniques pauvres qui sont sémantiquement incomplètes, car ils ne comprennent pas de description intentionnelle. Nous proposons, dans cette section, un descripteur sémantique complet, qui considère le service comme une entité unique aux multiples dimensions (dimension *intentionnelle*, *technique* et *contextuelle)*, qui s'intègrent dans une description sémantique unique. Chacune de ces dimensions est décrite plus en détails dans les sections suivantes.

6.3. La dimension intentionnelle d'un service

Un service est décrit avec une *intention principale* qu'il est capable de satisfaire. Celle-ci représente l'intention décrite par la communauté dans un domaine précis et qui va être analysée lors du processus de découverte de services (*cf.* Chapitre 7). D'un domaine à un autre, une même intention peut être exprimée de différentes manières. Cette *variabilité d'expression de l'intention* dépend de la communauté d'expert qui a établi l'ontologie d'intention et qui partage une vision commune de leur domaine, comme les ontologies soutenues par la communauté proposés par Mirbel et Crescenzo (Mirbel et Crescenzo, 2010). Ainsi, nous avons trouvé intéressant de prendre en considération cette variabilité dans notre descripteur de services.

De plus, comme nous l'avons souligné dans la section 3.4.3.3, une intention peut être *atomique ou agrégat*. Une *intention atomique* représente une intention qui ne se décompose pas en d'autres sous-intentions, tandis qu'une *intention agrégat* représente une intention qui se compose d'un ensemble de sous-intentions, qui à leur tour peuvent être décomposables.

Nous introduisons, dans la section 6.3.1, notre extension de OWL-S pour inclure les informations sur l'intention principale du service et sa variabilité d'expression. Ensuite, nous détaillons dans la section 6.3.2, notre prise en compte de la composition intentionnelle dans notre descripteur de services.

6.3.1. Le service et son intention principale

L'information relative à l'intention est décrite, comme l'illustre la Figure 33, dans une ontologie « *Intention Service* ». Cette ontologie représente une description de l'intention principale que le service est censé satisfaire. Elle se base sur une ontologie de domaine qui est établie par une communauté d'expert partageant un même domaine. Cette vision s'intègre parfaitement avec les SIP, puisque les services offerts par ces systèmes doivent s'adapter à une communauté d'utilisateurs spécifique.

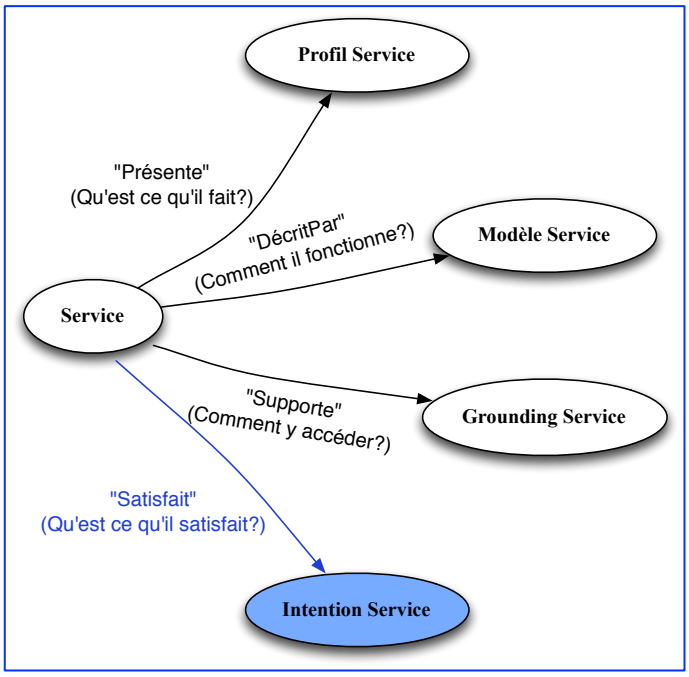


Figure 33. Extension de OWL-S pour représenter l'intention du service

Notre extension intentionnelle rajoute à OWL-S une nouvelle propriété au service, nommée « *satisfait* » et sa classe respective « *IntentionService* » (*cf.* Figure 33). Chaque instance de « *service* » permet ainsi de satisfaire une description intentionnelle de celui-ci. L'ontologie « *Intention Service* » fournit les informations nécessaires pour découvrir le service approprié qui satisfait au mieux une ou plusieurs intention(s) particulière(s). L'objectif ici est de mentionner l'*intention qu'un service permet de satisfaire* de manière à ce que ce service puisse être retrouvé à partir de l'intention de l'utilisateur.

La prise en compte de cet aspect intentionnel dans la description de services conduit à des modifications dans l'*ontologie de services,* pour faire référence à l'ontologie d'intention de services, et dans l'*ontologie de profil*, pour faire appel à l'ontologie d'intention lors du mécanisme de découverte de services par exemple. Nous introduisons dans les sections suivantes ces modifications ainsi que la nouvelle ontologie d'intention.

6.3.1.1. Extension de l'ontologie de service

Les parties constituant la description d'un service en OWL-S (*cf.* Figure 33) sont organisés dans une ontologie. Cette ontologie représente la structure de cette description exposant ces trois parties de base, à savoir le *profil*, le *modèle* et le *grounding* du service. Nous y rajoutons une quatrième partie, représentant la dimension intentionnelle d'un service.

Nous étendons ainsi cette ontologie de services, comme l'illustre la Figure 34, afin de mettre en avant le lien entre un *service* et une *intention*. Cette extension a été établie par l'ajout d'une classe *« Service Intention »* qui fournit une superclasse à la description intentionnelle du service (ligne 7-10 de la Figure 34).

```
5        <!-- Service Intention -->
6
7        <owl:Class rdf:ID="ServiceIntention">
8            <rdfs:label>ServiceIntention</rdfs:label>
9            <rdfs:comment>See comments above</rdfs:comment>
10       </owl:Class>
11
12       <!--   Sarisfies an Intention-->
13
14       <owl:ObjectProperty rdf:ID="satisfies">
15           <rdfs:domain rdf:resource="&service;#Service"/>
16           <rdfs:range rdf:resource=";#ServiceIntention"/>
17           <owl:inverseOf rdf:resource="#satisfiedBy"/>
18       </owl:ObjectProperty>
19
20       <owl:ObjectProperty rdf:ID="satisfiedBy">
21           <rdfs:comment>
22            There are no cardinality restrictions on this property. That is,
23            the same service intention can be used by many different services.
24           </rdfs:comment>
25           <rdfs:domain rdf:resource="#ServiceIntention"/>
26           <rdfs:range rdf:resource="&service;#Service"/>
27            <owl:inverseOf rdf:resource="#satisfies"/>
28       </owl:ObjectProperty>
```

Figure 34. Extension de l'ontologie de services

La classe *« Service Intention »* ne mentionne aucune représentation des services, mais elle impose les informations de base permettant de lier une intention avec une instance de service. Il existe une relation réciproque entre un *service* et une *intention de service*, de sorte qu'un service peut être lié à une intention et une intention à un service. Ces relations sont exprimées par les propriétés « *satisfait » (satisfies)* et « *satisfaitPar » (satisfiedBy)*. D'une part, la propriété *« satisfait »* décrit une relation entre une instance de service et une instance d'intention (ligne 14-18 de la Figure 34), indiquant ainsi que le service est capable de satisfaire cette intention. D'autre part, la propriété *« satisfaitPar »* représente l'inverse de *« satisfait »* (ligne 20-28 de la Figure 34). Elle spécifie qu'une intention donnée peut être satisfaite par un service.

6.3.1.2. Extension de l'ontologie de profil

L'ontologie de profile, comme nous l'avons mentionné dans la section 3.4.2.1, exprime *ce que le service réalise*. Cette ontologie donne une description de haut niveau d'un service, à

des fins de description, de publication et de découverte des services. Ainsi, afin d'inclure l'aspect intentionnel de notre descripteur de services, nous avons dû étendre l'ontologie du profil de services, comme l'illustre la Figure 35.

Dans la description du *« Profil Service »,* nous avons introduit un pointeur vers la sous-ontologie qui présente la description intentionnelle de service (ligne 44-47 de la Figure 35). Pour cela, nous proposons la propriété *« has_intention »*. Celle-ci a comme *domaine* la classe *profil* (*Profile)* et comme classe respective *(range)* la classe *« intention »*.

```
29      <owl:Ontology rdf:about="">
30        <owl:imports>
31            <owl:Ontology rdf:about="&profile;" />
32            <owl:Ontology rdf:about="&intention;" />
33        </owl:imports>
34      </owl:Ontology>
35
36      <!-- The context condition of the service -->
37
38      <owl:DatatypeProperty rdf:ID="context">
39        <rdfs:domain rdf:resource="&profile;#Profile"/>
40        <rdfs:range rdf:resource="&xsd;#anyURI"/>
41      </owl:DatatypeProperty>
42
43        <!-- has_intention is a pointer the process that is associated with the service -->
44      <owl:ObjectProperty rdf:ID="has_intention">
45        <rdfs:domain rdf:resource="&profile;#Profile"/>
46        <rdfs:range rdf:resource="&intention;#Intention"/>
47      </owl:ObjectProperty>
48      <owl:FunctionalProperty rdf:about="#has_intention"/>
```

Figure 35. Extension de l'ontologie de description de profil de service

La classe *intention* représente une spécialisation de la classe de profil de service *(Service Profile)*. Cette nouvelle classe utilise le caractère flexible et extensible de OWL-S. En effet, nous y ajoutons une nouvelle classe tout en gardant en perspective la possibilité de représenter l'ontologie d'intention de différentes manières.

```
5       <service:Service rdf:ID="EDITION_PROPOSAL_SERVICE">
6           <service:presents rdf:resource="#EDITION_PROPOSAL_PROFILE"/>
7           <service:describedBy rdf:resource="#EDITION_PROPOSAL_PROCESS_MODEL"/>
8           <service:supports rdf:resource="#EDITION_PROPOSAL_GROUNDING"/>
9           <service:satisfies rdf:resource="#PREPARE_PROPOSAL_INTENTION"/>
10      </service:Service>
11
12      <profile:Profile rdf:ID="EDITION_PROPOSAL_PROFILE">
13          <service:isPresentedBy rdf:resource="#EDITION_PROPOSAL_SERVICE"/>
14          <profile:serviceName xml:lang="en">
15              EditionProposalService
16          </profile:serviceName>
17          <profile:textDescription xml:lang="en">
18              This service is used to edit a proposal
19          </profile:textDescription>
20          <profile:has_process rdf:resource="#EDITION_PROPOSAL_PROCESS" />
21          <eprofile:has_intention rdf:resource="#PREPARE_PROPOSAL_INTENTION" />
22          <eprofile:context rdf:resource="http://www.crinfo.univ-paris1.fr/
23          ExtensionOWL-S/ContextDescription.xml#condition1"/>
24          ...
25      </profile:Profile>
26
27      <intention:Intention rdf:ID="EDITION_PROPOSAL_INTENTION">
28          <!-- Description of the service Intention -->
29      </intention:Intention>
```

Figure 36. Description du service *« Edition_Proposal_Service »* en OWL-SIC

Pour mieux comprendre cette description intentionnelle de services proposée à travers OWL-SIC, nous présentons, dans la Figure 36, un exemple de description de service. Dans cet exemple, nous présentons la description d'un service d'*édition de proposition* nommé *« Edition_Proposal_Service » (*ligne 5). Dans cette description, le service satisfait l'intention « *prepare proposition* » (ligne 9). Cette ligne représente l'identifiant de la description intentionnelle du service qui est détaillée dans le bloc de description des intentions (ligne 27-29). De plus, dans la partie de description de profile du service, nous rajoutons un pointeur vers la description de cette intention (ligne 21), afin de l'explorer dans un processus de découverte de services.

On observe que le changement que nous avons apporté au profil de service est minimal, puisque l'intention n'y est pas détaillée, étant décrite par l'ontologie d'intention. Ceci est particulièrement important pour maintenir une certaine compatibilité entre cette nouvelle description de services et les mécanismes de découverte.

6.3.1.3. Ajout d'une ontologie d'intention

Nous proposons, dans le cadre de cette extension, une ontologie d'intentions, laquelle propose une description des intentions qu'un service est capable de satisfaire. Nous considérons qu'un service satisfait une *intention principale*, qu'elle soit atomique ou composite. Cette intention est formulée selon le modèle de Prat (Prat, 1997) (*cf.* section 3.4.3.1). Ce modèle présente l'intention sous forme de *verbe,* de *cible* qui peut être un *objet* ou un *résultat* et un ensemble de *paramètres* qui sont optionnels. Les paramètres représentent la *façon*, la *quantité*, la *qualité*, la *direction*, la *qualité* et le *bénéficiaire*. La Figure 37 illustre les classes et les propriétés qui composent cette description intentionnelle d'un service.

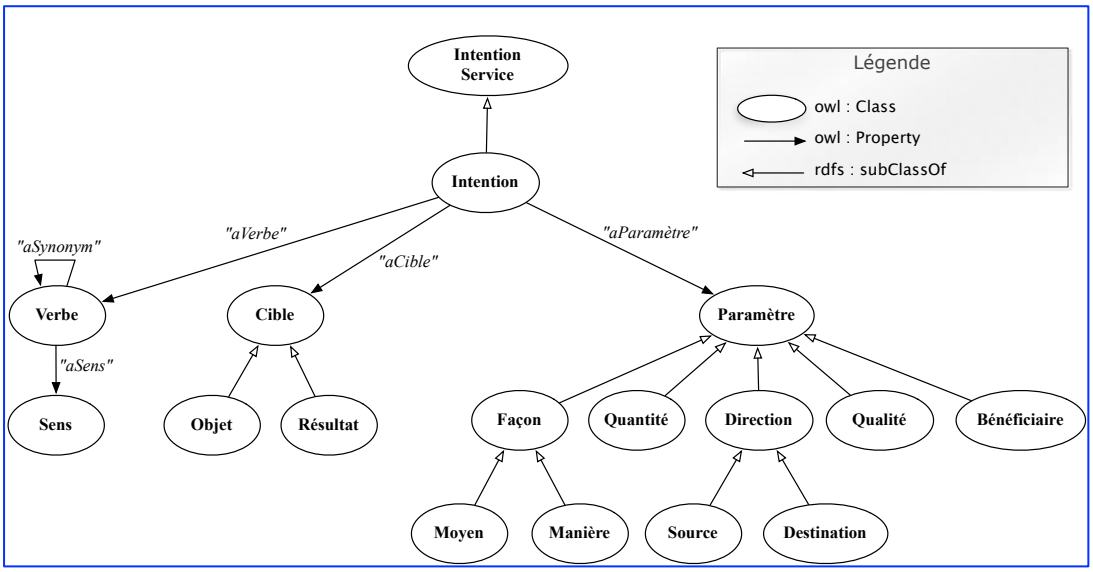


Figure 37. Les classes et les propriétés de l'ontologie d'intention

Selon Prat (Prat, 1997), le *verbe* expose l'action permettant la réalisation de l'intention. La *cible* représente soit l'*objet* existant avant la réalisation de l'intention ou le *résultat* découlant

de la satisfaction de l'intention. Les *paramètres* sont utiles pour clarifier l'intention et pour exprimer les informations supplémentaires. Le paramètre *direction* caractérise la source ou la destination des entités. D'un côté, la *destination* identifie l'emplacement des entités produites par la satisfaction de l'intention. De l'autre côté, la *source* identifie l'emplacement initial des entités. En outre, ce modèle d'intention représente le paramètre *façon*, faisant référence à l'instrument de la satisfaction de l'intention. Il représente le *moyen* ou la *manière* de satisfaire l'intention. Le *moyen* indique l'entité qui sert d'instrument pour atteindre l'intention, tandis que la *manière* identifie une approche dans laquelle l'intention peut être satisfaite. Enfin, le paramètre de *qualité* définit une propriété qui doit être atteinte ou maintenue.

Dans leurs travaux, Corby et al. (Corby et al., 2009) proposent une ontologie d'intention qui organise l'intention par *section*, selon le modèle de la carte (Rolland et al., 1998). Dans nos travaux, nous nous concentrons sur la description de l'intention elle-même. Toutefois, nous pouvons dégager certains points communs entre les deux ontologies, que nous avons entouré dans notre ontologie d'intention à la Figure 38. Par exemple, dans les deux ontologies nous suivons la même description des *paramètres* qui sont la *façon*, la *quantité*, la *qualité*, la *direction*, la *qualité*, etc. Ainsi, nous attachons dans ces deux travaux une *cible* à l'intention, appelée *objet* dans l'ontologie proposée par Corby et al. (Corby et al., 2009). Par contre, la description des *verbes* dans notre ontologie d'intention est plus riche. En effet, nous exploitons plus d'éléments autour du verbe, tels que le synonyme et le sens, que dans l'ontologie de Corby et al. (Corby et al., 2009). Notre ontologie d'intention représente une description sémantique du modèle d'intention de Prat (Prat, 1997), alors que l'ontologie de Corby et al. représente une description sémantique du modèle de la carte.

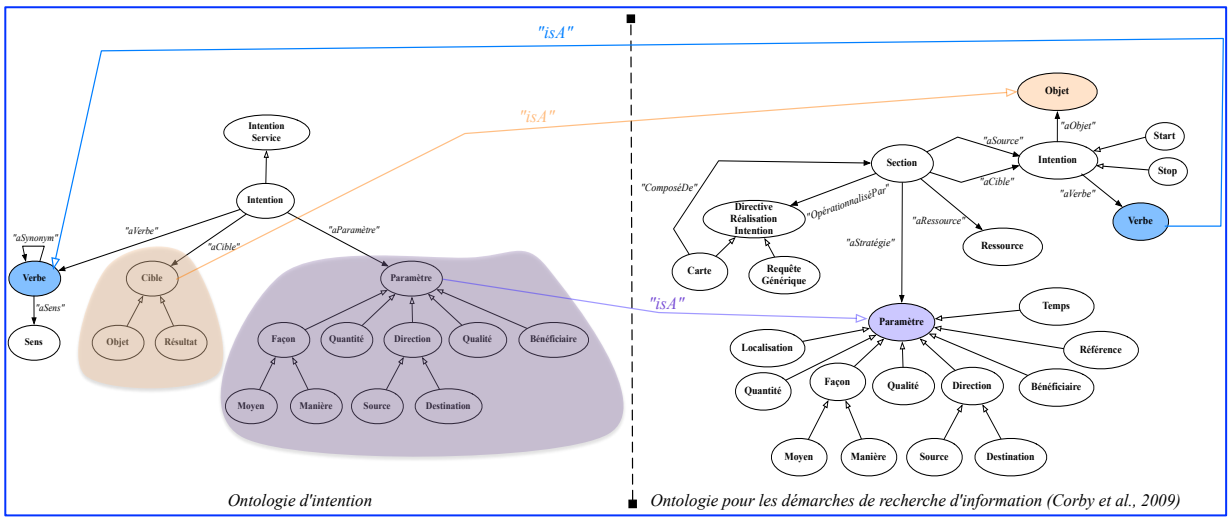


Figure 38. Comparaison avec l'ontologie d'intention de Corby et al. (Corby et al., 2009)

Chaque élément de cette description intentionnelle doit être lui-même sémantiquement défini au préalable dans des ontologies d'intentions qui sont dépendantes du domaine. En réalité, ces ontologies intentionnelles décrivent chacune un élément de l'intention : une o*ntologie de verbes*, une *ontologie de cibles* rendues accessibles par le SIP et des ontologies de paramètres. Les ontologies de verbes et de cibles établissent de manière non-ambiguë la

sémantique des *actions acceptées par le SIP dans l'espace de services*, représentant le verbe de l'intention, et l'ensemble *des cibles atteignables par le biais de cet espace*. Une telle définition, basée sur une ontologie prédéfinie, n'est envisageable que dans le cadre fermé d'un Système d'Information. En effet, ces systèmes n'autorisent pas un comportement ouvert sur des intentions et des cibles non-autorisées ou inconnues auparavant, de par leur importance stratégique dans les entreprises.

L'ontologie de verbe (*OntoV*) contient un ensemble de verbes liés à un domaine précis, celui du SIP, leurs significations et les relations entre ces verbes. *OntoV* établit une description sémantique des actions acceptées par les SIP. Elles présentent des *liens d'héritage* entre les verbes reflétant des relations de *hyponymie* (plus spécifique) et d'*hyperonymie* (plus général). A part ces deux liens, cette ontologie de verbes décrit la relation de *synonyme* entre deux verbes. Deux verbes *synonymes* représentent deux verbes identiques ou très voisins (ayant le même sens ou deux sens très proches). De plus, un *verbe* peut avoir un *sens* bien défini. En conséquence, nous décrivons le verbe, dans l'ontologie (*OntoV*), avec *son sens*. Le sens du verbe est introduit par le fournisseur de ce service afin de rajouter plus d'expressivité au verbe. En d'autres termes, il n'est là que pour aider l'utilisateur à mieux comprendre la signification des verbes qui existent dans l'ontologie des verbes (*OntoV*). Par exemple, le verbe « *consulter* » peut être décrit par plusieurs sens, à savoir *« examiner quelque chose »* ou *« interroger quelque chose »*.

L'ontologie des cibles (*OntoT*) représente des concepts sémantiques relatifs aux cibles pouvant être utilisées avec une intention. Cette ontologie établit de manière non-ambiguë la sémantique de l'ensemble des cibles atteignables par le biais de l'espace de services. Elle présente une vue hiérarchique de plusieurs types de cibles du domaine en question, en présentant les liens de spécialisations et de généralisation entre elles.

```
<intention:Intention rdf:ID="PREPARE_PROPOSAL_INTENTION">

    <!—The principal Intention of the service —>

    <intention:PrincipalIntention rdf:ID="PREPARE_PROPOSAL_PRINCIPAL_INTENTION">
        <intention:Verb rdf:resource="http:// www.crinfo.univ-paris1.fr/ExtensionOWL-S
        /Intention.owl#concept.intention.verb.prepare">
            Prepare
        </intention:Verb>

        <intention:Target rdf:resource="http:// www.crinfo.univ-paris1.fr/ExtensionOWL-S/
         Intention.owl#concept.intention.target">
            <intention:Result rdf:resource="http:// www.crinfo.univ-paris1.fr/ExtensionOWL-S/
            Intention.owl#concept.intention.target.result.Proposal">
                Proposal
            </intention:Result>
        </intention:Target>
    </intention:Principalintention>

    .....

</intention:Intention>
```

Figure 39. Exemple de description intentionnelle d'un service d'édition de proposition

La Figure 39 illustre un exemple de description intentionnelle du service d'*édition de proposition*. Ce service satisfait l'intention principale *« prepare proposition »* (ligne 5). Cette intention se compose du verbe « *prepare »* (ligne 6-9). La *cible* de l'intention représente le *résultat* obtenu par la satisfaction de cette intention, à savoir *« proposal »* (ligne 11-18).

Comme nous l'avons souligné précédemment, une même intention peut être exprimée de différentes manières selon la communauté. Ceci exprime la *variabilité dans l'expression de l'intention* que nous introduisons dans notre description intentionnelle des services. Pour un même service, nous pouvons associer différentes intentions, dont une est représentée comme *principale* et les autres comme ses *alternatives* dans des domaines différents. Par exemple, le service d'*édition de proposition* a comme intention principale *« prepare proposition »*. Cette intention est décrite dans le cadre d'une entreprise qui peut configurer ses propositions (*e.g.* logiciels) selon les appels d'offre. Cette même intention peut être exprimée différemment dans le cadre d'une autre entreprise qui propose ses produits sans configuration à ses clients. Dans ce cas, l'intention qui sera associée à ce service sera plutôt « *propose product »*. Ces multiples variations d'intention sont décrites dans le bloc de la description intentionnelle de la même manière que l'intention principale est décrite dans la Figure 39.

En plus de cette variabilité dans l'expression, une intention associée à un service peut être soit une intention simple, appelée intention *atomique*, soit une intention décomposable en plusieurs sous intentions, appelée intention *agrégat* (*cf.* section 3.4.3.3). Nous estimons qu'il est important de prendre en compte cette composition intentionnelle qui peut être exploitée dans d'éventuel mécanisme de composition de services. Nous discutons cette composition intentionnelle dans la section suivante.

6.3.2. La composition intentionnelle

Un service satisfait une intention particulière. Cette intention, et comme nous l'avons introduit précédemment, peut être *atomique* ou *agrégat*. Une *intention atomique* représente une intention simple et non décomposable, tandis que l'*intention agrégat* représente une intention qui est composée d'autres intentions. La satisfaction d'une intention agrégat nécessite la satisfaction de toutes ses sous intentions. Cet aspect intentionnel reflète la composition des intentions introduites dans les travaux de Kaabi et Souveyet (Kaabi et Souveyet, 2007) et Rolland et *al.* (Rolland et al., 2010). Néanmoins, et comme nous l'avons souligné dans la section 3.4.3.3, la vision de ces auteurs ne prend pas en considération l'évolution de la technologie de services qui peut supporter des logiciels avec des fonctionnalités réutilisables, ainsi que des systèmes hérités *(legacy system)* avec des processus complexes cachés par des technologies. Dans leurs travaux, Kaabi et Souveyet (Kaabi et Souveyet, 2007) et Rolland et al. (Rolland et al., 2010) considèrent que seuls les *services intentionnels atomiques* peuvent être opérationnalisés à travers le service logiciel. Ceci limite la réutilisation des systèmes hérités, puisque ces systèmes peuvent être associés à des intentions agrégats, mais ils ne peuvent pas être assimilés à des intentions atomiques simples.

Dans notre descripteur, nous nous sommes inspirés de ces travaux (Kaabi, 2007) (Rolland et al., 2010) et nous admettons ainsi l'importance de la composition intentionnelle dans la description d'un service. Par contre, et dans la perspective de suivre cette évolution technologique et de prendre en considération des systèmes complexes tels que les systèmes hérités, nous estimons qu'un *service atomique peut satisfaire des intentions complexes qui peuvent être agrégats*. Ainsi, puisqu'un *service peut être assez complexe en soi*, d'un point de vue intentionnel, nous avons décidé d'inclure la composition intentionnelle dans la description de services. Ceci nous distingue également de WSMO qui ne gère pas cet aspect.

En effet, un service englobe un certain processus intentionnel lequel peut être aussi simple que la satisfaction d'une seule intention atomique, mais qui peut être également assez complexe encapsulant une véritable composition intentionnelle. Que le service soit simple ou composite, l'intention qu'il satisfait peut être également atomique ou agrégat. Nous avons donc deux compositions orthogonales, représentées sur deux dimensions séparées : une *technique* et l'autre *intentionnelle*.

Dans le cas d'une composition traditionnelle, l'*agrégation technique* fournit des éléments techniques nécessaires à l'exécution du service. Par contre, l'*agrégation intentionnelle* permet de mieux comprendre, du point de vue utilisateur final, le service et les différentes façons de satisfaire son intention principale. Ainsi, nous proposons d'étendre le modèle de processus de OWL-S en incluant la spécification d'un *processus intentionnel*. Cette nouvelle spécification est décrite indépendamment de l'agrégation traditionnelle, étant donné que cette dernière dépasse le cadre de cette thèse. Notre extension est inspirée de la spécification technique des processus de services décrits en OWL-S et de la composition de services intentionnels proposée par (Kaabi et Souveyet, 2007) et (Rolland et al., 2010).

La Figure 40 présente notre extension de OWL-S pour inclure l'aspect de composition intentionnelle. Cette extension considère deux types de processus : le *processus intentionnel atomique* et le *processus intentionnel agrégat*. Le *processus intentionnel atomique* représente la spécification d'une intention qui est indécomposable. Ce processus intentionnel atomique est ainsi décrit uniquement par son intention principale qui n'est pas décomposable. Par exemple, le service *« LaunchConnexionVPN_service »* est un service qui permet de répondre à l'intention principale de lancement de connexion I5 = *{#connecter, #VPN, ∅}*. Cette intention peut être satisfaite directement et n'a pas besoin d'être décomposée en sous-intentions pour être satisfaite. A l'opposé, le *processus intentionnel agrégat* est un processus dont l'intention principale peut être décomposée en des intentions plus fines qui contribuent à sa satisfaction. Par exemple, le service *« AccessClientView_Service »* est un service qui répond à l'intention principale I6 = *{#consulter, #fiche_client, ∅}*. Cette intention se compose d'autres intentions : I6.1 = *{#connecter, #VPN, ∅}*, I6.2 = *{#afficher, #liste_client, ∅}* et I6.3 = *{#sélectionner, #client, ∅}*. Ces intentions représentent les intentions qui devront être satisfaites pour que l'intention principale le soit.

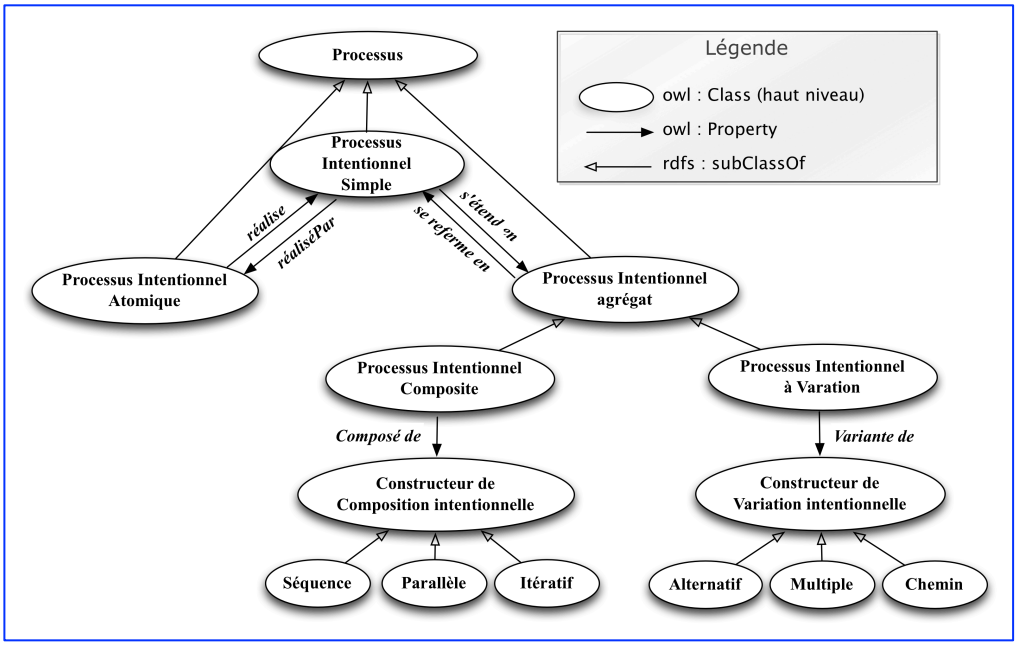


Figure 40. Extension de OWL-S pour prendre en considération la composition intentionnelle des services

Par ailleurs, notre extension du modèle de processus considère un *processus intentionnel simple*, lequel fournit une vue abstraite d'un processus intentionnel pouvant être atomique ou agrégat. Comme l'illustre la Figure 40, un *processus intentionnel simple* est réalisé par un processus intentionnel atomique et s'étend en un processus intentionnel agrégat. Un *processus intentionnel agrégat* peut être soit un *processus intentionnel composite* (qui établit des liens ET entre les intentions et affine l'intention principale en sous intentions qui lui sont associées), soit un *processus intentionnel à variante* (lequel établit des liens OU entre les intentions et offre le choix dans la satisfaction de l'intention principale) (*cf.* section 3.4.3.3). Ces processus, comme l'illustre la Figure 40, relient les intentions entre elles par des *constructeurs* bien spécifiques. Ces constructeurs, appelé *constructeur de composition intentionnelle* et *constructeur de variation intentionnelle*, sont inspirés des liens de choix et de composition proposés par Kaabi (Kaabi, 2007). Ces liens répondent parfaitement aux liens de composition et de variation qui peuvent exister entre les intentions.

Dans la cadre du *processus intentionnel composite,* le *lien séquentiel* a été choisi parce qu'il représente, dans le cadre de notre processus intentionnel composite, le cas le plus typique dans lequel il y a un ordre séquentiel entre les intentions qui contribuent à la satisfaction de l'intention principale. Nous avons choisi de représenter le *lien parallèle* parce qu'il reflète le cas dans lequel la satisfaction de l'intention principale nécessite la réalisation de certaines intentions en parallèle. Finalement, le *lien itératif* a été choisis parce qu'il existe des cas où l'intention principale ne peut être satisfaite que par l'itération de certaines de ces intentions composites. Dans la cadre du *processus intentionnel à variation*, nous avons choisi de représenter le *lien multiple* parce qu'il reflète un choix non exclusif dans la manière de satisfaire l'intention principale par les intentions composites. Parmi ces intentions composites, au moins une sera choisie. De plus, nous avons décidé de garder le *lien alternatif* parce qu'il présente le cas exigeant un choix exclusif dans la manière de satisfaire l'intention principale

par les intentions composites. Parmi ces intentions composites, uniquement une sera choisie. Nous gardons également le *lien chemin* parce qu'il offre un choix dans la façon d'atteindre l'intention principal. Dans ce cas, la *variation intentionnelle* porte sur un chemin d'intentions qui représente un ensemble d'alternatifs entre elles.

```
1   <eprocess:CompositeIntenionalProcess rdf:ID="PREPARE_PROPOSAL_INTENTIONAL_COMPOSITE_PROCESS">
2       ...
3
4       <eprocess:ComposedOf>
5           <eprocess:Sequence>
6               <process:Components rdf:parseType="Collection">
7
8                   <eprocess:AtomicIntenionalProcess rdf:ID="LAUNCH_VPN_CONNECTION_ATOMIC_INTENTIONAL_PROCESS">
9
10                  <eprocess:AtomicIntenionalProcess rdf:ID="ENCRYPT_DATA_ATOMIC_INTENTIONAL_PROCESS">
11
12                  <eprocess:AtomicIntenionalProcess rdf:ID="EDIT_PROPOSAL_ATOMIC_INTENTIONAL_PROCESS">
13
14                  <eprocess:VariableIntenionalProcess rdf:ID="SEND_PROPOSAL_VARIANT_INTENTIONAL_PROCESS">
15              </process:Components>
16          </eprocess:Sequence>
17      </eprocess:ComposedOf>
18
19  </eprocess:CompositeIntenionalProcess>
```

Figure 41. Exemple d'un processus intentionnel composite

Afin d'illustrer la composition intentionnelle dans la description de services, nous considérons le service « *Edition_ Proposal_Service* » qui permet de décrire une proposition commerciale et de l'envoyer au client. La Figure 41 décrit le processus intentionnel associé à ce service. Il représente un *séquencement* entre quatre processus intentionnels (ligne 5-16) dont l'ordre de composition doit être pris en compte lors de la satisfaction de l'intention principale. Plus précisément, d'un point de vue intentionnel, cette intention principale se compose de trois processus intentionnels atomiques, à savoir les *intentions atomiques « Launch Connection »* (ligne 8), *« Encrypt Data »* (ligne 10) et *« Edit Proposal »* (ligne 12). Celles-ci contribuent à la satisfaction directe de l'intention principale associée au service « *Edition_ Proposal_Service* ». De plus, celui-ci, et toujours d'un point de vue intentionnel, comporte une variante représentée par le processus intentionnel à variation décrit par l'intention *« Send Proposal »* (ligne 14).

```
22  <eprocess:VariantIntenionalProcess rdf:ID="SEND_PROPOSAL_VARIANT_INTENTIONAL_PROCESS">
23
24      ...
25
26      <eprocess:VariantOf>
27          <eprocess:Multiple>
28              <process:Components rdf:parseType="Collection">
29
30                  <eprocess:AtomicIntenionalProcess rdf:ID="SEND_MAIL_ATOMIC_INTENTIONAL_PROCESS">
31
32                  <eprocess:AtomicIntenionalProcess rdf:ID="SEND_FAX_ATOMIC_INTENTIONAL_PROCESS">
33
34              </process:Components>
35          </eprocess:Multiple>
36      </eprocess:VariantOf>
37
38  </eprocess:VariantIntenionalProcess >
```

Figure 42. Exemple de processus intentionnel à variation

Cette intention *« Send Proposal »*, comme l'illustre la Figure 42, représente un processus intentionnel présentant un choix non exclusif dans la manière de satisfaire cette intention par les deux intentions *« Send Mail »* (ligne 30) et *« Send Fax »* (ligne 32). Parmi ces deux intentions composites, au moins une devra être satisfaite. Ces deux dernières représentent deux processus intentionnels simples. Cet exemple (Figure 42) montre que pour satisfaire l'intention principale d'envoi de proposition, nous avons deux choix, soit l'envoi de la proposition par Mail **et/ou** l'envoi de la proposition par Fax.

Dans le cadre du processus intentionnel à variation, nous croyons que la variabilité sur la réalisation de l'intention peut dépendre de certains facteurs externes. Ces facteurs concernent des *informations contextuelles*. Tel un service, chaque variante peut avoir des conditions contextuelles dans lesquelles elle est la plus appropriée à être satisfaite. Ainsi, nous attribuons, pour chaque variante à l'intérieur d'un processus intentionnel, une description des conditions contextuelles (*cf.* section 6.4). Cette description de contexte représente les circonstances dans lesquelles il est le plus approprié de satisfaire une intention à variation plutôt qu'une autre.

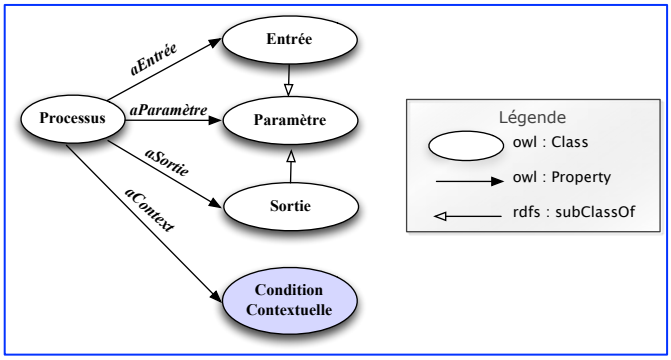


Figure 43. Extension de OWL-S pour décrire les ressources d'un service intentionnel et les conditions contextuelles

Afin de considérer l'influence du contexte sur les variantes intentionnelles, nous étendons le processus de services de OWL-S pour inclure les informations contextuelles dans la description de la variabilité du processus intentionnel mis en place par le service, comme l'illustre la Figure 43. Nous associons ainsi les conditions contextuelles pour chaque variante de ce processus. Cette description contextuelle permettra de rendre le plus explicite possible le processus intentionnel mis en place par le service. Il s'agit d'une démarche descriptive ouvrant de nouvelles possibilités à des mécanismes de découverte et de composition de services. Ainsi, une telle extension pourrait aider à choisir la variante en fonction des conditions de contexte. Nous étendons le modèle de processus OWL-S en intégrant les conditions contextuelles, comme nous allons le décrire dans la section 6.4.

Grâce à notre extension OWL-SIC, nous permettons une description de la composition intentionnelle, du point de vue utilisateur final. Cette extension expose la variabilité représentant les différentes manières de satisfaire les intentions de l'utilisateur. Cette description de la composition intentionnelle ouvre la porte pour des processus de composition

de services guidée par l'intention et le contexte, présenté à un haut niveau d'abstraction. Même si un tel processus dépasse le cadre de cette thèse, nous jugeons important que le descripteur OWL-SIC puisse permettre ces évolutions.

Après avoir introduit notre extension de OWL-S pour inclure l'aspect intentionnel des services, nous détaillons dans la section suivante le deuxième volet de cette extension laquelle inclut l'aspect contextuel.

6.4. La dimension Contextuelle d'un service

Une intention que l'utilisateur souhaite satisfaire émerge dans un contexte donné. Cette constatation soulève l'étroite relation entre la notion de contexte et celle d'intention. Cette relation représente notre troisième hypothèse décrite dans le Chapitre 1 et doit ainsi s'exprimer dans la description de services afin de mieux satisfaire les utilisateurs en leur proposant des services qui s'adaptent à leur contexte. Nous croyons que l'intention de l'utilisateur devient moins significative si nous ne la prenons pas avec son contexte d'usage. Ceci est dû à l'influence que peut avoir le contexte sur la satisfaction de l'intention. En effet, nous croyons que le contexte joue un rôle important dans le choix de la meilleure réalisation pour satisfaire l'intention de l'utilisateur (*cf.* hypothèse 4 dans la section 1.3).

Dans notre formalisation de l'*espace de services*, nous avons définit deux types de contexte auxquels un service est associé (*cf.* section 5.3.2.1) : le *contexte dans lequel se place et s'exécute un service* (Cx) et le *contexte requis représentant les conditions contextuelles* dans lesquelles le service est le plus apte à atteindre ses objectifs (CxR). Le contexte Cx sert non seulement à indiquer les conditions dans lesquelles le service est exécuté par son fournisseur, mais également à caractériser le positionnement de ce service dans l'espace de services. Tandis que, le contexte requis CxR représente les conditions de contexte permettant au service une meilleure possibilité de satisfaction des intentions qui lui sont associées.

Nous proposons ainsi d'étendre le langage de description de services OWL-S, en incluant, d'une part, la description des conditions contextuelles dans lesquelles un service est valide et exécutable (CxR), et d'autre part, le contexte dans lequel le service se place et s'exécute (Cx). Cette extension est introduite dans la description du profil de service *« Service Profile »*. Ceci est parce que le profil de services représente la partie de la description qui décrit *ce que fait le service* et comprend également une *description des exigences* que le demandeur de service doit satisfaire pour utiliser le service avec succès. Ainsi, étant donné que les conditions contextuelles d'un service (CxR) doivent correspondre au contexte de l'utilisateur afin de sélectionner ce service dans le processus de découverte de services (*cf.* Chapitre 7), nous concluons que ces conditions contextuelles doivent faire partie de ces exigences à l'intérieur du profil de services. De plus, puisqu'un service est décrit par un contexte (Cx) qui reflète les conditions dans lesquelles il s'exécute, nous concluons que ce contexte peut être décrit également dans cette partie afin de refléter non seulement ce que fait le service mais également dans quelles conditions il l'a fait.

Selon Najar et al. (Najar et al., 2011c), une condition contextuelle (CXR) peut être considérée comme faisant partie de la description du service, car elle indique des situations auxquelles le service est mieux adapté. Toutefois, selon Kirsch-Pinheiro et al. (Kirsch-Pinheiro et al., 2008), les informations de contexte (Cx) ne peuvent pas être statiquement enregistrées dans le profil de service puisque certaines de ces informations sont capturées dynamiquement. En effet, les propriétés de contexte liées à l'exécution du service peuvent évoluer dans le temps, alors que le profil de service est censé être une description statique plutôt stable d'un service.

```
<profile:Profile rdf:ID="EDITION_PROPOSAL_PROFILE">
    ....
    <eprofile:context rdf:resource="http://www.crinfo.univ-paris1.fr/
    ExtensionOWL-S/ContextDescription.xml"/>
    ...
</profile:Profile>
```

Figure 44. Inclusion de la description contextuelle dans la description du profil de services en OWL-S

Dans l'objectif d'inclure les informations de contexte, qui sont dynamiques, dans la description d'un service, nous adoptons l'approche proposée par Kirsch-Pinheiro et al. (Kirsch-Pinheiro et al., 2008). Ceci consiste à enrichir le profil de service de OWL-S par un *attribut de contexte*. Cet attribut introduit dans la description une URL qui fait référence à une ressource externe, laquelle contient la description de contexte. Prenons, par exemple, la description de contexte d'un service qui est contenue dans un fichier externe se trouvant à l'URI suivante : *http://www.crinfo.univ-paris1.fr/ExtensionOWL-S/ContextDescription.xml*. Notre extension de OWL-S revient à inclure, comme l'illustre la Figure 44, cette URI dans la description du profil à travers la balise <*eprofile : context*>. Cette alternative permet aux fournisseurs de services de mettre à jour plus facilement, voir d'une manière automatique, les informations de contexte sans pour autant modifier la description de service elle-même.

Afin d'arriver à cette description de contexte, il nous faut donc un modèle de contexte à suivre permettant de décrire comment ces informations de contexte peuvent être représentées. Dans le cadre de notre *espace de services*, nous avons présenté un méta-modèle de contexte (*cf.* section 5.3.1) représentant une modélisation générique de contexte Cx et qui peut être instancié par les différentes approches de modélisation de contexte analysées dans la section 2.3.2. Ainsi, afin de représenter notre description de contexte, nous instancions ce méta-modèle de contexte.

Dans les sections suivantes, nous présentons une instanciation de notre méta-modèle de contexte qui va être utilisée, par la suite, pour représenter le contexte Cx. En se basant sur cette modélisation, nous introduisons notre description des conditions contextuelles de CXR.

6.4.1. Le modèle de contexte

L'ensemble des sujets et des éléments de contexte observés dans un espace de services varient, bien évidemment, en fonction du système et du modèle de contexte mis en place. Depuis quelques années, une tendance se dégage sur les modèles de contexte les plus récents : la description sémantique de ces éléments (Najar et al., 2009). Comme nous l'avons conclu dans la section 2.3.2, de plus en plus de modèles utilisent les ontologies pour décrire ces éléments. Ces ontologies fournissent un vocabulaire représentant de la connaissance sur les informations de contexte selon le domaine (Wang et al., 2004) (Preuveneers et al., 2004) (Reichle et al., 2008). La structuration des informations contextuelles dans des ontologies a de multiples avantages. Ceci permet de partager une compréhension commune de la structure de l'information de contexte (Gu et al., 2004). De plus, la définition d'une ontologie assez riche, permet la réutilisation des connaissances représentées du domaine. Les ontologies ont également comme avantage aussi d'être extensible, permettant ainsi la réutilisation de concepts de base d'une ontologie et de l'étendre selon le domaine auquel elle s'applique. Finalement, la représentation explicite des concepts de contexte dans des ontologies permet également de raisonner sur l'ensemble de concepts. C'est pour toutes ces raisons que de plus en plus de modèles de contexte se basent sur les ontologies pour définir ces éléments.

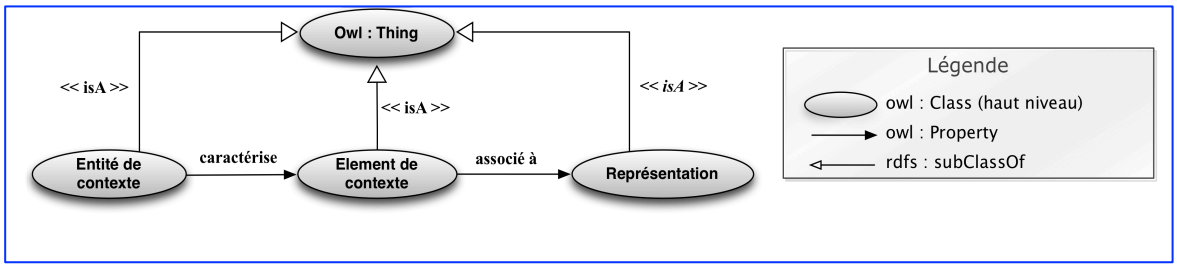


Figure 45. Concept de base de notre Modèle de contexte soutenu par une ontologie

Nous proposons, dans cette section, un modèle de contexte soutenu par une ontologie. Malgré, la multitude de modélisation de contexte sous forme d'ontologie, telle que SOUPA (Chen et al., 2004), COBrA (Chen et al., 2003), CoDaMos (Preuveneers et al., 2004), entre autres, nous proposons une autre modélisation de contexte qui prend en considération tous les éléments qu'on a mis en avant lors de notre méta-modélisation de contexte. Les concepts de base de notre ontologie de contexte sont illustrés à la Figure 45. Ce modèle de contexte représente une instanciation de notre méta-modèle de contexte, présentée dans la section 5.3.1, où les concepts du modèle de contexte sont dérivés et compatibles avec les éléments clés de notre méta-modèle. Plus concrètement, nous introduisons le concept *« entité de contexte »* comme une instanciation du concept *« sujet observé »,* nous gardons la même dénomination pour l'*« élément de contexte »* et finalement nous dérivons le concept *« représentation »* du concept *« métadonnée ».* L'ontologie de contexte que nous proposons se base sur le langage d'ontologie Web OWL (OWL - Web Ontologie Langage), ainsi tous ses concepts héritent des propriétés du concept *« owl : Thing »* qui représente la racine de la description OWL. Au cœur de cette ontologie, se trouve l'*entité de contexte* qui représente à qui/quoi l'information de contexte fait référence : *utilisateur, dispositif, etc.* De plus, cette

ontologie tourne autour des *éléments de contexte*, qui correspondent à tout élément détectable et calculable statiquement ou dynamiquement, tels que *température, localisation, etc.*

De plus, afin de fournir une ontologie extensible qui est bien structurée et facile à comprendre, intégrer et étendre, nous utilisons une ontologie multi-niveaux. Cette ontologie, basée sur l'ontologie de contexte MUSIC (Paspallis, 2009), est conçue pour être facilement extensible par des sous-ontologies, dépendantes du domaine, en fonction du système dans lequel le modèle s'applique. Ces ontologies sont construites par niveaux que nous pouvons étendre facilement. L'utilisation d'ontologies offre une description sémantique particulièrement riche et fournit de nouvelles perspectives pour les mécanismes d'adaptation grâce aux différentes possibilités de raisonnement. Ainsi, les concepts de base de notre méta-modèle sont décrits sémantiquement dans une ontologie multi-niveaux, qui étend les concepts de base de notre ontologie de contexte aux éléments les plus communs et les plus utilisés. A la Figure 46, nous représentons, à ce niveau, l'ensemble des concepts que nous avons identifiés comme couramment utilisés dans la littérature. Par exemple, « *GPS* » représente un héritage du concept « *localisation* » qui représente lui même une spécification du concept « *environnement* ». Cette ontologie multi-niveaux est structurée comme suit :

- ***Ontologie générique (Niveau supérieur*)** décrit les informations de base de contexte dans un environnement pervasif. Ces informations sont assez génériques et communes à plusieurs domaines. A ce niveau, les concepts de base sont définis. Trois concepts de niveau supérieur sont ainsi définis (voir Figure 46) : l'*utilisateur*, l'*environnement* et l'*entité informatique*. Ces concepts sont organisés autour du concept « *Element de Contexte* » qui représente le point d'entrée pour déclarer l'ontologie de haut-niveau. L'*Element de Contexte* est spécialisé par la suite afin d'y représenter les concepts les plus spécifiques relatifs aux notions de l'utilisateur, l'environnement et l'entité informatique. Cette ontologie de haut-niveau est en suite spécialisée à un niveau inférieur. En effet, dans la conception de cette ontologie, le niveau supérieur de l'ontologie générique peut par la suite être étendu selon le domaine en un ensemble d'ontologies de niveau inférieur ;
- ***Ontologie spécifique du domaine (Niveau inférieur)*** comporte des informations de contexte spécifiques à un domaine d'application. Le détail des informations de base de contexte, représentées dans le niveau supérieur de cette ontologie, est défini dans ce niveau inférieur qui est spécifique à un domaine donné et qui varie d'un domaine à un autre. En d'autres termes, les informations du concept représenté dans ce niveau de l'ontologie sont divisées en plusieurs sous-domaines, à l'instar du modèle de contexte proposé par Wang et al. (Wang et al., 2004). Chacun définit les détails et les propriétés spécifiques pour un domaine d'application donné. Par exemple, le concept *« localisation »* de l'ontologie de haut niveau peut être spécifié en *« localisation Architecturale »* qui est de même spécifiée en *« bâtiment »*, *« chambre »* et *« passage »*, entres autres. Dans un autre domaine, ce concept *« localisation »* peut être spécifié en *« localisation GPS »* qui a comme propriétés : *« longitude », « latitude »* et *« altitude ».*

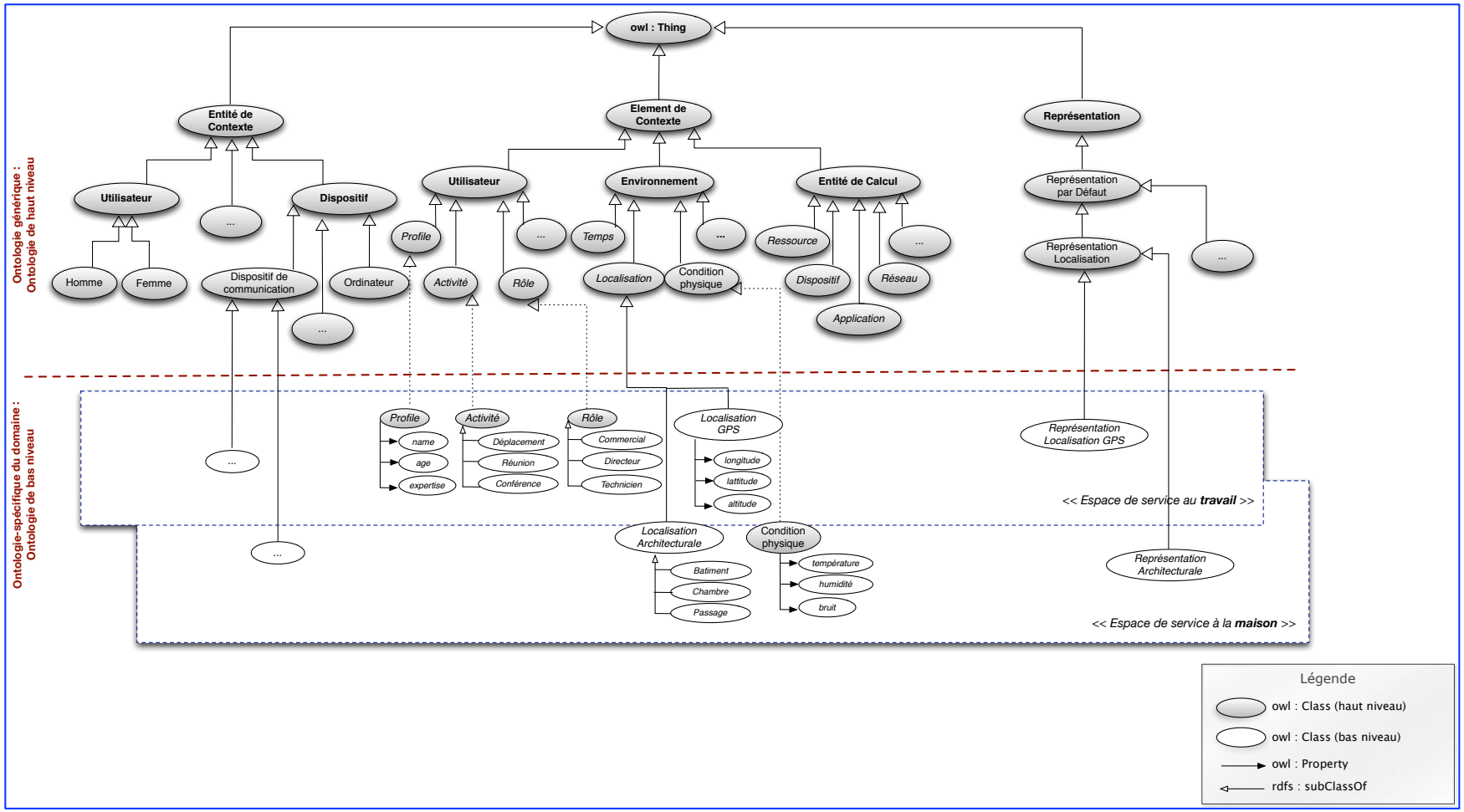


Figure 46. Ontologie multi-niveaux de contexte

A partir des éléments définis sur l'ontologie de haut niveau, nous pouvons distinguer trois catégories pour les éléments de contexte qui y sont représentés :

- *L'**utilisateur*** joue un rôle central au sein de l'environnement pervasif, puisque bien souvent les systèmes pervasifs cherchent à s'adapter à l'utilisateur. Cette adaptation à l'utilisateur représente d'ailleurs un des principes fondateurs de l'Informatique Pervasive, laquelle soutient que l'informatique doit s'adapter à l'utilisateur et non l'inverse. Les concepts de cette catégorie décrivent ainsi l'utilisateur et tout ce qui lui est rattaché. Ces éléments de contexte sont ainsi observés à partir d'un sujet représentant l'utilisateur. Parmi les concepts plus importants de cette catégorie, on peut citer : *profil*, *rôle*, *activité*, etc. ;
- *L'**environnement*** représente un élément clé dans la spécification de contexte puisque l'utilisateur est représenté dans un environnement avec lequel il interagit. L'environnement peut être de différentes natures : *physique* (GPS, température, localisation, etc.), *logique* (mémoire disponible sur le terminal, préférences de l'utilisateur, etc.) et *organisationnel* (rôle de l'utilisateur, état d'exécution d'un processus, etc.). Cette catégorie comporte des informations sur la *localisation*, le *temps*, et de manière plus générale, le *contexte physique* représentant les conditions environnementales. De manière générale, ces éléments de contexte correspondent à l'observation d'un environnement physique autour d'un sujet donné, le plus souvent l'utilisateur, même si d'autres entités seraient également possibles (la localisation d'un terminal ou d'un capteur, par exemple). ;
- *L'**entité de calcul*** représente la description du matériel et du logiciel dans l'environnement observé bien souvent, spécifique à un dispositif (sujet). Cela comprend la spécification du *t*ype du *réseau*, de la *bande passante* disponible, de la taille de la *mémoire* d'un dispositif, etc.

Cette modélisation va nous permettre de décrire le contexte dans lequel s'exécute le service ($\mathcal{Cx}$) (*cf.* section 6.4.2). Quant au contexte requis ($\mathcal{CxR}$) (*cf.* section 6.4.3), il va être structuré de la même manière que $\mathcal{Cx}$. Par contre, puisqu'il représente des conditions contextuelles, ceci nécessite un langage sémantique adéquat pour pouvoir représenter ces conditions. Ainsi, nous répartissons la description globale de contexte du service, qui est enregistrée dans le fichier externe, en deux parties. D'un côté, nous décrivons les informations de contexte qui se réfèrent aux conditions contextuelles ($\mathcal{CxR}$) dans lesquelles il est plus utile de sélectionner et appeler ce service. Cette partie sera décrite dans un bloc intitulé *« condition »*. De l'autre côté, nous décrivons le contexte dans lequel un service s'exécute ($\mathcal{Cx}$). Cette partie sera décrite dans un bloc intitulé *« state »* et fera référence aux conditions de contexte dans lesquelles un service est exécuté par le fournisseur de services (l'état dans lequel le service est exécuté).

Nous détaillons la description de contexte d'exécution $\mathcal{Cx}$ ainsi que le contexte requis $\mathcal{CxR}$ d'un service dans les sections suivantes.

6.4.2. Le contexte d'exécution d'un service (Cx)

Selon cette modélisation de contexte, présentée dans la section précédente, nous représentons le contexte d'exécution d'un service Cx dans une partie nommée *« state »* du fichier de description contextuelle. Ce contexte Cx est décrit comme un ensemble d'observations acquises par des capteurs (*cf.* section 5.3.2.1). Chaque observation est décrite par l'entité que nous avons observée, l'élément de contexte attribué à cette entité, ainsi que la valeur associée à cet élément et sa représentation.

Pour illustrer cette description de contexte Cx, nous penons l'exemple de l'intention *« prepare proposal »* satisfaite par le service *« edition proposal service »*. Ce service a été exécuté dans un contexte décrivant, entre autre, l'observation de la *taille de l'écran* de son *dispositif* qui était égale à *10 pouces* et l'observation de sa *localisation* qui était en *France*. Nous représentons, dans la Figure 47, l'observation de la localisation de l'utilisateur.

```
1   <ctx:state>
2     ...
3     <ctx:contextstate>
4           <ctx:hasEntity resource="http://127.0.0.1/services/
5           ContextModel.owl#concept.Entity.User"/>
6           <ctx:hasContextElement resource="http://127.0.0.1/services/
7           ContextModel.owl#concept.ContextElement.Environement.Localisation"/>
8           <ctx:contextValueSet>
9                   <ctx:contextValue>
10                      <ctx:hasContextElement resource="http://127.0.0.1/services/
11                      ContextModel.owl#
12                      concept.ContextElement.Environement.Localisation.City"/>
13                      <ctx:hasRepresentation resource="http://127.0.0.1/services/
14                      ContextModel.owl#concept.Representation.Device_Representation.
15                      City_Localisation_Representation"/>
16                      <ctx:valueSet>
17                          <ctx:valueElement>
18                              <ctx:hasContextElement resource="http://127.0.0.1/services/
19                              ContextModel.owl#concept.ContextElement.User.Localisation.City"/>
20                               <ctx:value> France </ctx:value>
21                          </ctx:valueElement>
22                      </ctx:valueSet>
23                  </ctx:contextValue>
24              </ctx:contextValueSet>
25      </ctx:contextstate>
26      ...
27  </ctx:state>
```

Figure 47. Exemple d'une observation de contexte du service Cx

Cette observation est décrite dans la partie *« context state »* (ligne 3-25 de la Figure 47). Elle décrit l'*élément de contexte localisation* (ligne 6-7) de l'*entité observé utilisateur* (ligne 4-5). Plus spécifiquement, cette observation capture la *ville* dans laquelle se trouve l'utilisateur (ligne 10-12). L'élément de contexte *ville* représente une spécification de l'élément de contexte *localisation* dans l'ontologie multi-niveaux de contexte (*cf.* section 6.4.1). Ceci représente l'association qui peut exister entre les éléments de contexte représentée dans notre méta-modèle de contexte (*cf.* section 5.3.1). Dans cette observation, la *valeur associée* à la localisation de l'utilisateur est *France* (ligne 20). Cette valeur est *représentée* sous la forme d'une localisation par ville (ligne 13-15).

Inspiré de cette modélisation et description contextuelle de Cx, nous décrivons dans la section suivante le contexte requis d'un service (CxR).

6.4.3. Le contexte requis par un service ($Cx\mathcal{R}$)

En se basant sur la modélisation de contexte Cx, nous représentons le contexte requis d'un service $Cx\mathcal{R}$ dans une partie nommée *« condition »* du fichier de description contextuelle. Ce contexte $Cx\mathcal{R}$ est décrit comme un ensemble de conditions contextuelles (*cf.* section 5.3.2). Chaque condition contextuelle est décrite par l'entité sur laquelle porte la condition, l'élément de contexte attribué à cette entité, ainsi que la condition. Cette condition est exprimée par un ensemble *d'opérateurs*. D'une part, ces opérateurs peuvent être des *opérateurs simple*, tels que l'*égalité*, la *différence*, l'*intervalle*, la *supériorité* et l'*infériorité*. D'autre part, ces opérateurs peuvent être plus *complexes*, par exemple un *opérateur permettant de déterminer si la localisation de l'utilisateur se trouve dans une zone particulière.* Il est à noter que certains de ces opérateurs nécessitent des opérations de *transformation* d'une représentation à une autre pour pouvoir évaluer la satisfaction de la condition.

```
1   <ctx:condition>
2     ...
3     <ctx:contextcondition>
4           <ctx:hasEntity resource="http://127.0.0.1/services/
5           ContextModel.owl#concept.Entity.User"/>
6           <ctx:hasContextElement resource="http://127.0.0.1/services/
7           ContextModel.owl#concept.ContextElement.Environement.Localisation"/>
8           <ctx:contextValueConditionSet>
9                   <ctx:contextValueCondition>
10                      <ctx:hasContextElement resource="http://127.0.0.1/services/
11                      ContextModel.owl#
12                      concept.ContextElement.Environement.Localisation"/>
13                      <ctx:hasRepresentation resource="http://127.0.0.1/services/
14                      ContextModel.owl#concept.Representation.Device_Representation.
15                      Localisation_Representation"/>
16                      <ctx:valueConditionSet>
17                          <ctx:valueElementCondition>
18                              <ctx:hasContextElement resource="http://127.0.0.1/services/
19                              ContextModel.owl#concept.ContextElement.User.Localisation"/>
20                               <ctx:operator> Location-In </ctx:value>
21                               <ctx:value> Europe </ctx:value>
22                          </ctx:valueElementCondition>
23                      </ctx:valueConditionSet>
24                  </ctx:contextConditionValue>
25              </ctx:contextValueConditionSet>
26      </ctx:contextcondition>
27      ...
28  </ctx:condition>
```

Figure 48. Exemple d'une condition de contexte CxR

Pour illustrer cette description de contexte $Cx\mathcal{R}$, nous penons le même exemple utilisé dans la section précédente. Notre objectif est de porter une condition qui permet d'évaluer la satisfaction de l'observation décrite dans la Figure 47. D'une manière générale, cette condition permet d'évaluer si la localisation de l'utilisateur (France) satisfait la condition qui indique que ce service exige que la localisation soit en Europe. Cette condition contextuelle est décrite dans la partie *« context condition »* (ligne 3-26 de la Figure 48). Elle porte sur l'*élément de contexte localisation* (ligne 6-7) de l'*entité utilisateur* (ligne 4-5). Plus spécifiquement, cette condition emploie un *opérateur « Location-In »* (ligne 20) sur une valeur de localisation qui est l'*Europe* (ligne 21). Ceci veut dire que la condition portée sur la

localisation observée par *Cx* doit être en Europe. Si cette localisation observée se trouve en Asie par exemple, alors cette condition du service ne sera pas satisfaite.

Dans ce cadre, nous rappelons que les conditions contextuelles d'une intention à variation, décrite dans la section 6.3.2, sont décrites de la même manière que le contexte $\mathcal{CXR}$ du service. D'une manière conceptuelle, un contexte requis $\mathcal{CXR}$ peut être attribué soit à un service soit à une intention à variation.

6.5. Conclusion

Nous avons proposé, dans le cadre de ce chapitre, un descripteur sémantique des services intentionnels et sensibles au contexte. Ce descripteur, à l'encontre des différentes approches présentées dans le Chapitre 3, propose une extension de OWL-S qui combine et exploite la relation entre le contexte et l'intention dans la description du service. Une telle description guidée par le contexte et l'intention est essentielle dans le cadre d'un SIP transparent et centré utilisateur, celui-ci doit se caractériser par son adaptabilité au contexte et sa compréhension de l'utilisateur et de ses besoins.

Cette description complète notre proposition de l'espace de services, puisqu'elle permet de décrire l'ensemble de services qu'il englobe. Ainsi, avec la description de l'ensemble des services disponibles dans cet espace, nous pouvons avoir une réelle visibilité et une description de celui-ci. De plus, ce descripteur intentionnel et contextuel sera exploité par les mécanismes de découverte (*cf.* Chapitre 7) et de prédiction (*cf.* Chapitre 8) de services puisqu'ils interagissent avec un répertoire de services décrit selon ce descripteur.

Cependant, il convient de souligner que les mécanismes de découverte et de prédiction, que nous allons présenter dans le Chapitre 7 et le Chapitre 8 n'exploitent pas, pour autant, toute la description intentionnelle et contextuelle que nous avons présentée dans ce chapitre. Ces deux mécanismes vont interagir notamment avec le *profil de service* de OWL-SIC, la partie de la description de services qui indique *ce que fait le service.* Cette partie comprend une *description de contexte et des intentions possibles* de ce service. C'est elle qui va être traitée dans le processus de découverte et de prédiction de services.

Ceci n'empêche pas que l'extension décrivant le processus intentionnel peut être exploitée par un mécanisme de composition de services lequel pourrait gérer la variabilité et la composition dans la satisfaction de l'intention de l'utilisateur, selon des contraintes contextuelles spécifiques. Cette partie est en dehors de la portée de notre travail de thèse et reste ici en tant que travaux futurs.

Chapitre 7. DECOUVERTE DE SERVICES GUIDEE PAR L'INTENTION ET LE CONTEXTE

7.1. INTRODUCTION

Aujourd'hui, nous constatons que l'utilisateur interagit avec une panoplie de dispositifs et de services offerts par l'ensemble des SI qui nous entourent. Alors que de grands efforts ont été concentrés sur la recherche sémantique et sur l'adaptation au contexte, surtout à la localisation et aux dispositifs utilisés, nous constatons aujourd'hui les limitations de ces approches, notamment une certaine surcharge de l'utilisateur dû aux *« faux-positifs »*. Les utilisateurs se voient proposer plusieurs implémentations pour un même service, sans avoir pour autant le bagage nécessaire pour comprendre ces implémentations, ce qui nuit à la transparence d'utilisation de ces systèmes. La clé du succès serait donc d'offrir à l'utilisateur le service qui satisfait ses besoins, sans qu'il soit forcé de comprendre des détails sur l'implémentation ou sur les contraintes des dispositifs utilisés.

Nous pensons que seulement une approche centrée sur l'utilisateur sera capable d'apporter des services adaptés au contexte d'utilisation tout en gardant un niveau de transparence convenable. C'est dans ce souci que nous proposons, dans le cadre de ce chapitre, un processus de découverte de services guidé non seulement par le contexte de l'utilisateur, mais également par son intention. Ce processus s'encadre donc dans une vision plus globale, celle avancée par la notion d'espace de services (*cf.* Chapitre 5), laquelle soutient une vision centrée utilisateur pour les SIP.

Tel que nous l'avons défini à travers la notion d'espace de services, une intention représente les exigences formulées par l'utilisateur, qui sait ce qu'il attend d'un service mais qui ne sait pas indiquer comment y parvenir. Ces intentions émergent dans un contexte d'utilisation précis. Il s'agit d'un élément important dans le processus d'adaptation d'un système, processus que nous souhaitons enrichir par la notion d'intention. Nous pensons non seulement que la satisfaction des intentions de l'utilisateur dans un SIP dépend du contexte dans lequel se trouve cet utilisateur, mais aussi que le contexte impacte la manière dont les intentions sont satisfaites, tout comme le contexte impacte le choix des services qui seront exécutés. Cette relation peut être exploitée dans la découverte de services : grâce à l'observation de l'intention et du contexte d'utilisation, un processus de découverte plus précis peut être mis au point, offrant aux utilisateurs les services les plus adaptés à leurs besoins.

Dans ce chapitre, nous présentons un processus de découverte de services à la fois contextuel et intentionnel. Ce processus se base sur un principe de mise en correspondance entre l'intention de l'utilisateur, formulée dans sa requête, et les intentions des services, d'une part, et entre le contexte courant de l'utilisateur et les conditions contextuelles associées aux services, d'autre part. Nous présentons, par la suite, une implémentation de ce mécanisme et

analysons les résultats des expérimentations menées notamment par rapport au passage à l'échelle, à la précision et au rappel. Ces trois critères nous semblent particulièrement importants, puisqu'ils démontrent non seulement la faisabilité de l'approche, mais également son intérêt pour une découverte plus appropriée aux utilisateurs.

7.2. PROCESSUS DE DECOUVERTE DE SERVICES GUIDE PAR L'INTENTION ET LE CONTEXTE

7.2.1. Principe

Un utilisateur interagit avec un SIP à travers l'espace de services. Cette interaction reflète le désir de l'utilisateur d'avoir un SIP qui soit capable de satisfaire ses besoins (formulés en termes d'intentions) selon son contexte d'usage. Ceci consiste à lui offrir, en toute transparence, le service le plus adapté à son contexte courant et le plus approprié à ses intentions. Ainsi, découle le besoin de présenter un nouveau processus de découverte de services qui prend en considération le contexte et l'intention de l'utilisateur lors de cette phase de découverte du service le plus approprié.

Par conséquence, nous proposons un processus de découverte de services guidée par l'intention et le contexte. Ce processus se base sur un algorithme de mise en correspondance *(matching)* des services. Cet algorithme va permettre de comparer sémantiquement, et en se basant sur un ensembles de mesures de similarité, l'intention et le contexte de l'utilisateur avec chaque service disponible dans le répertoire de services (*cf.* Chapitre 9). Dans ce cadre, le concept d'*intention* est utilisé pour exposer les services et mettre en œuvre une vision centrée utilisateur des SIP dans un contexte donné. Nous soutenons qu'une meilleure prise en compte de l'intention de l'utilisateur peut conduire à une meilleure compréhension de l'utilisation réelle des services, ce qui par conséquent peut améliorer la précision et le rappel des services choisis pour satisfaire les besoins des utilisateurs. Par ailleurs, les *informations contextuelles* jouent un rôle central dans ce processus de découverte de services car elles influencent le choix des meilleures stratégies de satisfaction de l'intention.

Ce processus de découverte est proposé afin de masquer la complexité de la mise en œuvre des services dans un environnement hétérogène et dynamique, et par conséquent d'atteindre la transparence promise et l'efficacité souhaitée des Systèmes d'Information Pervasifs (Najar et al., 2012a) (Najar et al., 2012b). En d'autres termes, ce processus est proposé afin d'assurer un certain niveau de transparence et d'efficacité des systèmes, en s'adaptant au contexte (*permettant ainsi une meilleure gestion de l'hétérogénéité*) et en prenant en considération l'intention (*permettant ainsi une meilleure compréhension de l'utilisateur et de l'utilisation réelle des services*). Ceci permet, par conséquent, d'améliorer la sélection des services en réduisant les services *« faux-positifs »* offerts à l'utilisateur.

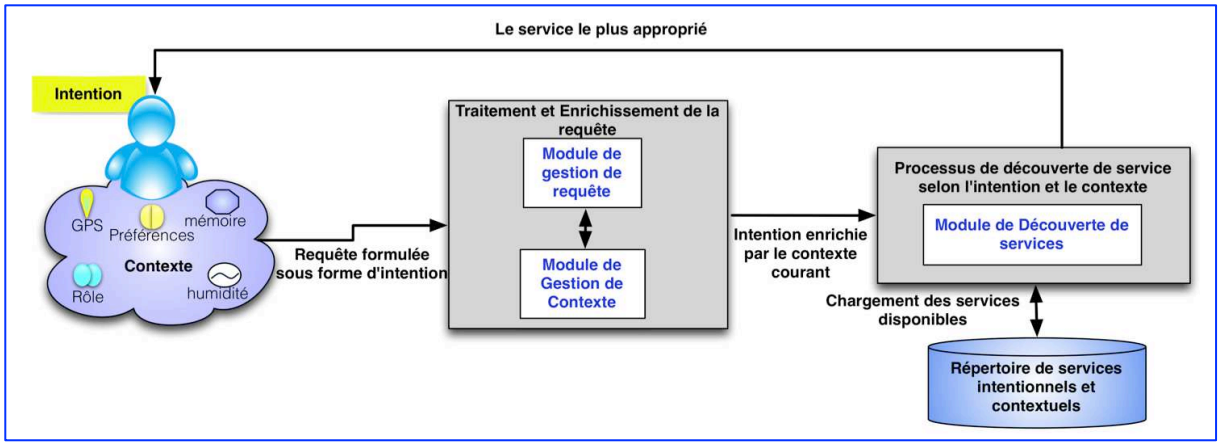


Figure 49. Interaction avec le module de découverte de services dans le cadre de l'architecture de gestionnaire de SIP

Dans l'architecture proposée dans le Chapitre 9, le *processus de découverte de services*, comme l'illustre la Figure 49, se déclenche lorsqu'il reçoit une demande de la part du *module de gestion de requête*. Plus précisément, lorsqu'un utilisateur envoie sa requête sous forme d'intention, cette requête est prise en compte par le *module de gestion de requête.* Ce dernier communique, ensuite, avec le *module de gestion de contexte* afin d'enrichir l'intention de l'utilisateur avec son contexte courant. Par la suite, il envoie cette requête enrichie au *module de découverte de services.* Dès lors, le *module de découverte de services* interagit avec le répertoire de services intentionnels et contextuels pour charger les services disponibles. Par la suite, il lance son *mécanisme de découverte pour trouver le service qui répond au mieux à l'intention et au contexte de l'utilisateur*.

Le processus de découverte de services, en fonction de ces deux concepts de contexte et d'intention, aidera les utilisateurs à découvrir le service le plus approprié pour eux. Ce processus représente celui qui répond aux *besoins immédiats* de l'utilisateur dans un contexte donné. Il utilise la description de services sémantiques, que nous avons présentée dans le Chapitre 6, dans un algorithme de découverte de services guidé par l'intention et le contexte, qui sera présenté dans la suite de ce chapitre. Cet algorithme effectue une mise en correspondance sémantique afin de sélectionner le service le plus approprié à l'utilisateur. Le but de cet algorithme est de classer les services disponibles en fonction de leurs informations contextuelles et intentionnelles. Il sélectionne, ensuite, celui qui répond au mieux à l'intention immédiate de l'utilisateur dans son contexte courant.

7.2.2. Algorithme de découverte de services guidée par le contexte et l'intention

Dans cette section, nous présentons l'algorithme de découverte de services selon une vision intentionnelle et contextuelle, comme l'illustre la Figure 50. Cet algorithme se compose d'une première phase de *mise en correspondance intentionnelle* (*cf.* section 7.2.2.3), qui permet de déterminer si l'intention du service correspond sémantiquement à l'intention de l'utilisateur, et d'une deuxième phase de *mise en correspondance contextuelle* (*cf.* section

7.2.2.4), qui permet de déterminer si la description de contexte associée à un service correspond à la description de contexte courant de l'utilisateur. Le résultat final représente le service qui répond au mieux à l'intention et au contexte de l'utilisateur.

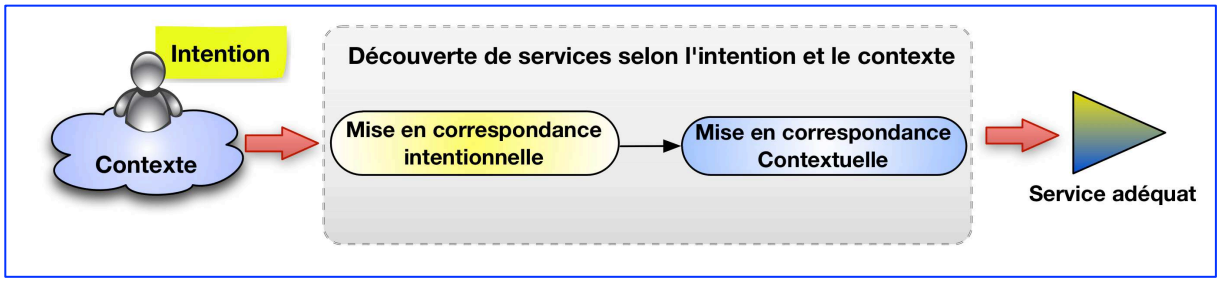


Figure 50. Principe de la découverte de services selon une vision intentionnelle et contextuelle

7.2.2.1. Concepts préliminaires : les mesures de similarités et la mise en correspondance sémantique

Une mesure de similarité représente une métrique qui calcule une distance entre deux éléments en se basant sur des fonctions plus ou moins adaptées au type des éléments utilisés. Parmi ces mesures de similarités, nous pouvons citer, par exemple :

- La *mesure de similarité euclidienne (d)* représente une mesure qui calcule la distance entre deux vecteurs *x* et *y* de dimension *n*, comme l'illustre la Formule 1.

$$d\,(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Formule 1. Mesure de similarité euclidienne

- La *mesure de similarité cosinus (θ)* représente une mesure qui permet de calculer, comme l'illustre la Formule 2, la similarité entre deux vecteurs *A* et *B* à *n* dimension en déterminant un angle *θ* entre eux. Cet angle s'obtient par le produit scalaire et la norme des vecteurs.

$$\theta\,(A,B) = arccos\frac{A\,.\,B}{||A||.||B||}$$

Formule 2. Mesure de similarité cosinus

- La *mesure de similarité sémantique (ds)* représente une mesure qui calcule la distance sémantique (normalisée) entre deux concepts en se basant sur leur lien sémantique dans une ontologie. Ce lien représente les relations de généralisation et de spécialisation entre les concepts. Cette distance est calculée selon la Formule 3, où *l* représente le nombre de liens qui séparent un concept *a* d'un concept *b*.

$$ds\ (a,b) = \frac{1}{(l+1)}$$

Formule 3. Mesure de similarité sémantique

Un certain nombre de travaux de recherche utilisant ces mesures ont été menés dans la découverte des services sémantiques, tel que nous l'avons détaillé dans la section 3.5.2. La plupart de ces travaux, tels que (Paolucci et al., 2002) et (Klusch et al., 2009), se basent sur le principe de la *mise en correspondance sémantique* entre deux concepts. Ceci correspond à une comparaison entre deux concepts en se basant sur leur niveau hiérarchique dans une ontologie. Plus précisément, la mise en correspondance sémantique se base sur la mesure de la similarité sémantique afin de calculer la distance sémantique qui sépare ces deux concepts. Cette mise en correspondance permet ainsi de déterminer la *relation de subsomption* entre deux concepts dans une ontologie. Cette relation de subsomption permet de lier des concepts spécifiques à des concepts plus génériques dans une ontologie.

Dans l'algorithme de mise en correspondance de services, que nous proposons dans la section suivante, nous appliquons différents types de mesures de similarité : (i) des *mesures de similarité sémantique entre les intentions* (entre les *cibles* décrites dans l'ontologie des cibles (*cf.* section 7.2.2.3.1) et entre les *verbes* dans l'ontologie des verbes (*cf.* section 7.2.2.3.2)) ; (ii) des *mesures de similarité sémantique entre les contextes* (entre les *entités*, les *éléments* et les *représentation de contexte* dans l'ontologie multi-niveaux de contexte (*cf.* section 7.2.2.4.1)) ; et (iii) d'autres types de *mesures de similarité appliqués essentiellement sur les valeurs des éléments de contexte observés* (*cf.* section 7.2.2.4.1).

7.2.2.2. L'algorithme de mise en correspondance de services guidée par le contexte et l'intention

D'une manière générale, l'algorithme de mise en correspondance que nous proposons se décompose en quatre étapes, comme indiqué schématiquement dans la Figure 51.

Pour chaque service disponible, l'intention de l'utilisateur est d'abord comparée à l'intention principale associée à ce service (étape 1.1). Ensuite, le contexte courant de l'utilisateur est comparé, sémantiquement et en se basant sur des mesures de similarités, aux conditions de contexte associées au service $(C\mathcal{X}\mathcal{R})$ (étape 1.2). Finalement, le score du degré de mise en correspondance final entre la requête de l'utilisateur (représentant son intention enrichie par son contexte courant) et chacun des services disponibles dans le répertoire de services est calculé (étape 1.3). Ce score va définir le classement de chaque service répondant à l'intention et au contexte de l'utilisateur. À partir de ce classement, le service qui obtient le score le plus élevé est proposé à l'utilisateur.

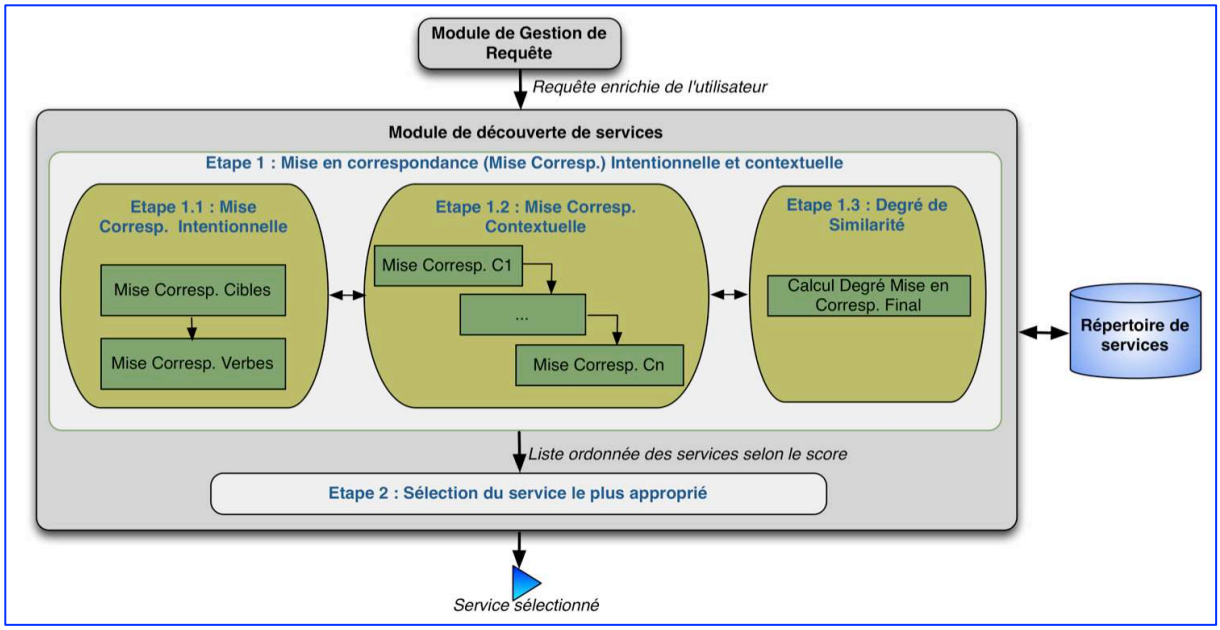


Figure 51. Aperçu du processus de découverte de service guidée par le contexte et l'intention

Plus en détails, nous présentons, dans la Figure 52, notre algorithme de mise en correspondance de services. Cet algorithme prend en entrée une *intention* présentant le besoin de l'utilisateur (I_u), une description du *contexte courant* de l'utilisateur (Cx_u) et un ensemble de *services disponibles* (Sv) dans le répertoire de services intentionnels et contextuels, décrit en OWL-SIC (*cf.* Chapitre 6). De plus, il prend en entrée un ensemble d'ontologies, à savoir une *ontologie de cibles* ($OntoT$), une *ontologie de verbes* ($OntoV$) (*cf.* section 6.3.1.3) et une *ontologie multi-niveaux de contexte* ($OntoCX$) (*cf.* section 6.4.1). L'algorithme de mise en correspondance produit à la sortie, une liste qui contient les couples $<sv_i, S_{score}>$. Le *service* sv_i représente le service candidat qui peut répondre à l'intention de l'utilisateur dans son contexte. Le *degré de similarité* S_{score} représente le degré de similarité entre l'intention et le contexte du service sv_i et de l'utilisateur $\mathcal{U}$.

Procedure Service Discovery (Cx_u, I_u, Sv, $OntoT$, $OntoV$, $OntoCX$)

(1) $S_{ranked} = \emptyset$ /* *Liste des services classés avec leur score de mise en correspondance* */
(2) $I_{score} = 0$ /* *Degré de similarité entre* I_u *et l'intention du service* (I_{sv_i}) */
(3) $C_{score} = 0$ /* *Degré de similarité entre* Cx_u *et le contexte du service* (Cx_{sv_i}) */
(4) $S_{score} = 0$ /* *Score final de la mise en correspondance entre* (I_u, Cx_u) *et* (I_{sv_i}, Cx_{sv_i}) */

(5) **For each** $sv_i \in Sv$ **Do**
(6) $Cx_{sv_i} = GetContext\ (sv_i)$
(7) $I_{sv_i} = GetIntention\ (sv_i)$
(8) $I_{score} = IntentionMatching\ (I_u, I_{sv_i}, OntoT, OntoV)$ /* *Mise en correspondance intentionnelle* */
(9) **If** $I_{score} > \alpha$ **Then**
(10) $C_{score} = ContextMatching\ (Cx_u, Cx_{sv_i}, OntoCX)$ /* *Mise en correspondance contextuelle* */

(11) **If** $C_{score} > \beta$ **Then**
(12) $S_{score} = (I_{score} + C_{score})/2$
(13) $S_{ranked} = S_{ranked} \cup \{<sv_i, S_{score}>\}$
(14) **End If**
(15) **End If**
(16) **End For**
(17) $sv_{selected} = GetBestService\ (S_{ranked})$ /* Retourne le service ayant le score le plus élevé */
(18) **Return** $sv_{selected}$
(19) **End Procedure**

Figure 52. Algorithme de découverte de services basée sur l'intention et le contexte

Dans cet algorithme, nous précisons deux paramètres α et β qui prennent leurs valeurs dans l'intervalle [0,1]. Le *paramètre* α représente le seuil au-delà duquel l'algorithme procède à la mise en correspondance contextuelle. Si le degré de similarité des intentions est supérieur ou égal au seuil α, alors l'algorithme lance la mise en correspondance entre les contextes. Le *paramètre* β représente le seuil au-delà duquel l'algorithme procède au calcul du score global de la mise en correspondance et de l'ajout du service parmi la liste des services candidats. Si le degré de similarité des descriptions de contexte est supérieur ou égal au seuil β, alors l'algorithme considère ce service comme service candidat potentiel répondant au contexte de l'utilisateur. Le paramètre α permet de configurer de manière indirecte la distance d'expression de l'intention de l'utilisateur avec celle associée à un service. Le seuil β, quant à lui, permet de paramétrer la distance d'expression du contexte de l'utilisateur avec le contexte requis du service.

La configuration d'un seuil α très élevé, proche de un, implique inévitablement que l'utilisateur connaît exactement la formulation des intentions associées aux services et l'utilisateur ne peut pas être dans un mode exploratoire ou dynamique. Par contre, la configuration d'un seuil α très bas, permet un meilleur mode exploratoire mais peut conduire à la sélection de services qui n'intéressent pas l'utilisateur. Ainsi, il est important de bien définir le seuil de paramétrage α en fonction de l'usage qu'on souhaite faire du système.

Un seuil β très élevé correspond à un couplage fort entre la description contextuelle des services et celle du contexte courant de l'utilisateur. Ce couplage fort entraîne un coût de maintenance élevé lorsque le modèle de contexte évolue ou freine la réutilisation potentielle des services en fonction d'un contexte. Par contre, un seuil β très bas peut engendrer la sélection d'un service inapproprié au contexte. C'est pour cette raison qu'il faut bien spécifier le seuil de paramétrage β en fonction de la complexité de la description du modèle de contexte et de son niveau de généralisation/spécialisation.

Ces deux paramètres α et β sont définis au préalable par le concepteur du système de manière empirique en fonction des scénarios. Ainsi, le concepteur est amené à tester l'algorithme avec son propre jeux de tests en utilisant son répertoire de services et ses ontologies. A chaque fois, il doit modifier les seuils et évaluer les résultats obtenus. Ceci va

lui permettre de trouver le seuil le plus approprié sélectionnant les résultats les plus pertinents. Ainsi, l'avantage d'un tel seuil de paramétrage est son adaptation au cas par cas. Toutefois, cette évaluation empirique demande un effort supplémentaire du concepteur.

L'algorithme de mise en correspondance, comme l'illustre la Figure 52, commence par sélectionner un service (sv_i) parmi les services disponibles dans le répertoire (ligne 5). Ensuite, le contexte (Cx_{svi}) et l'intention (I_{svi}) du service (sv_i) sont récupérés (ligne 6-7). Après avoir récupéré toutes ces informations, l'intention (I_{svi}) est mise en correspondance sémantiquement avec l'intention de l'utilisateur (I_u). Le degré de similarité résultant est par la suite stocké dans la variable I_{score} (ligne 8). Le score I_{score} est par la suite comparé au seuil α (ligne 9). Si le score est inférieur au seuil alors l'algorithme ne traite pas la mise en correspondance contextuelle et passe directement au service suivant. Ceci est parce que le service en cours de traitement n'est pas capable de satisfaire l'intention de l'utilisateur. Ainsi, même si ce service peut correspondre au contexte courant de l'utilisateur, il ne sera pas sélectionné puisqu'il ne répond pas au besoin de l'utilisateur. Ce choix va nous permettre de gagner en performance en évitant un traitement (mise en correspondance contextuelle) inutile. Par contre, si le degré de similarité I_{score} est supérieur ou égal au seuil α alors le contexte de l'utilisateur (Cx_u) est évalué par rapport au contexte requis du service (CXR_{svi}). Le degré de similarité résultant est par la suite stocké dans la variable C_{score} (ligne 10). Le score C_{score} est ensuite comparé au seuil β (ligne 11). Si le score est inférieur au seuil alors l'algorithme ne sélectionne pas le service comme service candidat et passe directement au service suivant. Dans le cas contraire, le degré de similarité final (S_{score}) est, par la suite calculé selon la formule suivante $S_{score} = (I_{score} + C_{score}) / 2$ (ligne 12). Par la suite, le service et son degré de similarité sont enregistrés dans la liste des services candidats (S_{ranked}) (ligne 13). Finalement, si un autre service est toujours disponible, alors l'algorithme relance la même démarche décrite ci-dessus. Sinon, l'algorithme sélectionne le service ($sv_{selected}$) ayant le score le plus élevé (ligne 17) et le retourne comme résultat final (ligne 18).

Les sections suivantes détaillent chacune des étapes de l'algorithme de mise en correspondance de services guidé par l'intention et le contexte.

7.2.2.3. La mise en correspondance intentionnelle des services

La correspondance intentionnelle *(Intention Matching)* représente la première étape de l'algorithme de mise en correspondance guidée par l'intention et le contexte. Celle-ci consiste à comparer sémantiquement l'intention de l'utilisateur (I_U), qui représente son besoin lors de sa demande d'un service, et l'intention principale (I_{svi}) qu'un service peut satisfaire.

Dans le cadre de ce processus de découverte de services, nous formulons l'intention selon le modèle de Prat (Prat, 1997). Nous avons choisi de représenter l'intention sous forme de *verbe* et de *cible* puisqu'ils représentent les éléments de base d'une intention et qui sont obligatoires lors de sa représentation. Nous estimons que la prise en compte des paramètres dans cette étape peut nuire à la qualité des résultats obtenus. Par exemple, Frioui (Frioui,

2012) a proposé une solution pour prendre en compte les paramètres d'une intention dans l'algorithme de découverte de services que nous avons proposé. Pour ce faire, cet auteur ne prend pas tous les paramètres, mais uniquement le paramètre *façon* qu'il juge obligatoire et les deux paramètres *référence* et *qualité* qu'il juge optionnel. Par la suite, il propose de comparer syntaxiquement ces paramètres lors de l'étape de la mise en correspondance intentionnelle. Notre analyse de cette approche a soulevé certaines complexités à savoir : « *comment trouver les paramètres significatifs ? », « comment gérer un nombre élevé de paramètres de types différents ? », ou encore « comment prendre en compte ces paramètres sans nuire aux résultats obtenus ? ».* Ainsi, nous pouvons conclure que la prise en compte des paramètres dans la mise en correspondance intentionnelle ne va pas obligatoirement améliorer les résultats obtenus. Bien au contraire, le nombre de paramètres de type différent à prendre en compte rend la description de l'intention bien trop précise et pas assez générale. De plus, en prenant en compte ces paramètres lors de cette phase, les services retournés comme les plus appropriés vont être trop spécifiques. En conséquence, ceci peut nuire à la qualité des résultats obtenus, essentiellement à la précision et au rappel. C'est pour cette raison que nous avons estimer, à ce stade de travail, que le gain obtenu de cette approche n'est pas suffisamment satisfaisant comparé à la complexité rajoutée à l'algorithme de découverte.

Ainsi, la mise en correspondance intentionnelle, énoncée dans la Formule 4, se base sur deux étapes :

- La *mise en correspondance des cibles* (*Target Matching)* permettant de déterminer le degré de similarité entre les cibles de l'intention de l'utilisateur et celle du service ;
- La *mise en correspondance des verbes* (*Verb Matching)* permettant de déterminer le degré de similarité entre les verbes de l'intention de l'utilisateur et celle du service.

$$IntentionMatching(I_U, I_S) = \begin{array}{c} \forall T_U, \exists T_S : TargetMatching(T_U, T_S) \\ + \\ \forall V_U, \exists V_S : VerbMatching(V_U, V_S) \end{array}$$

Formule 4. Mise en Correspondance Intentionnelle

Chacune de ces relations, aussi bien pour le verbe que pour les cibles, correspond à un score allant de 0 (quand il n'existe aucune relation) à 1 (quand il existe une relation exacte). Ce score est calculé en se basant sur la mesure de similarité sémantique décrite dans la section 7.2.2.1. Ainsi, le score correspondant aux relations intermédiaires est calculé en fonction du nombre des niveaux hiérarchiques qui séparent les concepts dans l'ontologie.

Dans le cadre de notre mise en correspondance intentionnelle, nous avons choisi de commencer par la mise en correspondance des cibles et ensuite la mise en correspondance des verbes. Même si un verbe caractérise l'action à effectuer sur une cible particulière, celle-ci joue un rôle discriminent dans la description de l'intention. Ainsi, en commençant la mise en correspondance intentionnelle par la comparaison des cibles, nous délimitons notre recherche

à un sous-ensemble de cibles (la cible en question et à ses similaires). Selon nos expérimentations, la prise en compte de la mise en correspondance des cibles avant celle des verbes, lors de l'exécution, améliore sensiblement le temps de réponse de notre algorithme. C'est pour cela que nous avons choisi d'analyser les cibles en premier lieu.

Plus en détails, nous présentons dans la Figure 53 notre algorithme de mise en correspondance intentionnelle comparant l'intention de l'utilisateur avec celle du service, afin de déterminer si l'intention du service en question peut répondre au besoin de l'utilisateur. Cet algorithme prend en entrée une *intention présentant le besoin de l'utilisateur* (I_u), une *intention principale satisfaite par un service* (I_{svi}), une *ontologie de cibles* ($Onto\mathcal{T}$) et une *ontologie de verbes* ($Onto\mathcal{V}$). Il produit à la sortie le *degré de similarité* (I_{score}) entre I_u et I_{svi}.

Procedure Intentional Matching (I_u, I_{svi}, $Onto\mathcal{T}$, $Onto\mathcal{V}$)

(1) $I_{score} = 0$ /* Degré de similarité entre I_u et l'intention du service (I_{svi}) */
(2) $\mathcal{T}_{score} = 0$ /* Degré de similarité entre la cible de I_u ($\mathcal{T}_u$) et la cible de I_{svi} ($\mathcal{T}_{vi}$) */
(3) $\mathcal{V}_{score} = 0$ /* Degré de similarité entre le verbe de I_u ($\mathcal{V}_u$) et le verbe de I_{svi} ($\mathcal{V}_{vi}$) */

(4) $\mathcal{V}_u$ = *GetVerb* (I_u)
(5) $\mathcal{T}_u$ = *GetTarget* (I_u)
(6) $\mathcal{V}_{svi}$ = *GetVerb* (I_{svi})
(7) $\mathcal{T}_{svi}$ = *GetTarget* (I_{svi})
(8) $\mathcal{T}_{score}$ = *TargetMatching* ($\mathcal{T}_u$, $\mathcal{T}_{svi}$, $Onto\mathcal{T}$) /* Mise en correspondance des cibles */

(9) **If** $\mathcal{T}_{score} > \alpha$ **Then**
(10) $\mathcal{V}_{score}$ = *VerbMatching* ($\mathcal{V}_u$, $\mathcal{V}_{svi}$, $Onto\mathcal{V}$) /* Mise en correspondance des verbes */
(11) **If** $\mathcal{V}_{score} > \alpha$ **Then**
(12) $I_{score} = (\mathcal{T}_{Score} + \mathcal{V}_{score})/2$
(13) **End If**
(14) **End If**

(15) **Return** I_{score}
(16) **End Procedure**

Figure 53. Algorithme de la mise en correspondance intentionnelle

L'algorithme de mise en correspondance débute par l'extraction du verbe ($\mathcal{V}_u$) et de la cible ($\mathcal{T}_u$) de l'intention de l'utilisateur (I_u) (ligne 4-5). Ensuite, le verbe ($\mathcal{V}_{svi}$) et la cible ($\mathcal{T}_{svi}$) sont déterminés à partir de l'intention du service (I_{svi}) (ligne 6-7). La cible $\mathcal{T}_u$ est mise en correspondance sémantiquement avec la cible $\mathcal{T}_{svi}$ (ligne 8). Le degré de similarité résultant est stocké dans la variable $\mathcal{T}_{score}$. Ce score $\mathcal{T}_{score}$ est par la suite comparé au seuil α (ligne 9). Si le score est inférieur au seuil, alors l'algorithme ne traite pas la mise en correspondance des verbes et se termine avec un $I_{score} = 0$. Par contre, si le degré de similarité $\mathcal{T}_{score}$ est supérieur ou

égal au seuil α alors le verbe ($\mathcal{V}_u$) est mis en correspondance sémantiquement avec le verbe ($\mathcal{V}_{svi}$) (ligne 10). Le degré de similarité résultant est stocké dans la variable $\mathcal{V}_{score}$. Ce score $\mathcal{V}_{score}$ est, à son tour, comparé au seuil α (ligne 11). Si le score est inférieur au seuil, alors l'algorithme ne continue pas le traitement du service puisqu'il ne réponds pas à l'intention de l'utilisateur, et renvoie un I_{score} égale à zéro (ligne 15). Sinon, le degré de similarité final des intentions (I_{score}) est, par la suite, calculé selon la formule suivante $I_{score} = (\mathcal{V}_{score} + \mathcal{T}_{score}) / 2$ (ligne 12). L'algorithme retourne, à la fin de cette procédure, le score I_{score} calculé (ligne 19).

Dans la suite de cette section, nous présentons la mise en correspondance des cibles et celles des verbes.

7.2.2.3.1. La mise en correspondance des cibles

La mise en correspondance des cibles *(Target Matching)* représente la première étape de la mise en correspondance intentionnelle. Cette étape se base sur la mise en correspondance sémantique entre les cibles, selon une ontologie de cibles ($Onto\mathcal{T}$) (*cf.* section 6.3.1.3), afin de déterminer le degré de similarité entre elles.

L'ontologie des cibles ($Onto\mathcal{T}$) représente des concepts sémantiques relatifs aux cibles pouvant être utilisées avec une intention. Elle décrit les différentes cibles rendues accessibles par le SIP. Cette ontologie établit de manière non-ambiguë la sémantique de l'ensemble des cibles atteignables par le biais de l'espace de services. Cette ontologie présente une vue hiérarchique de plusieurs types de cibles du domaine en question, en présentant les liens de spécialisations et de généralisation entre eux.

Dans ce chapitre, nous avons choisi d'illustrer notre processus de découverte de services dans le domaine du *voyage* en raison de sa simplicité, permettant ainsi de mieux explorer le processus. La Figure 54 illustre un extrait de l'ontologie de cibles dans ce domaine. Cette ontologie comporte cinq principaux éléments, à savoir :

- *Hébergement (accommodation)* correspond à des hébergements de types *hôtel (hotel)*, *Bed and Breakfast*, *camping*, etc. ;
- *Activité (activity)* correspond à des activités d'*aventure (adventure)*, de *relaxation (relaxation)*, de *sport (sports)*, etc. ;
- *Contact (contact)* correspond à des *cartes* (*card*) ou à des *routes (road)* ;
- *Destination (destination)* correspond à des destinations vers la *plage (beach)*, des *zones urbaines (urban area)*, des *zones sportives (sports destination)*, etc. ;
- *Météo (weather)* correspond à la *sécheresse (drought)*, l'*éclair (lightning)*, la *météo de la saison (weather season)*, etc. ;

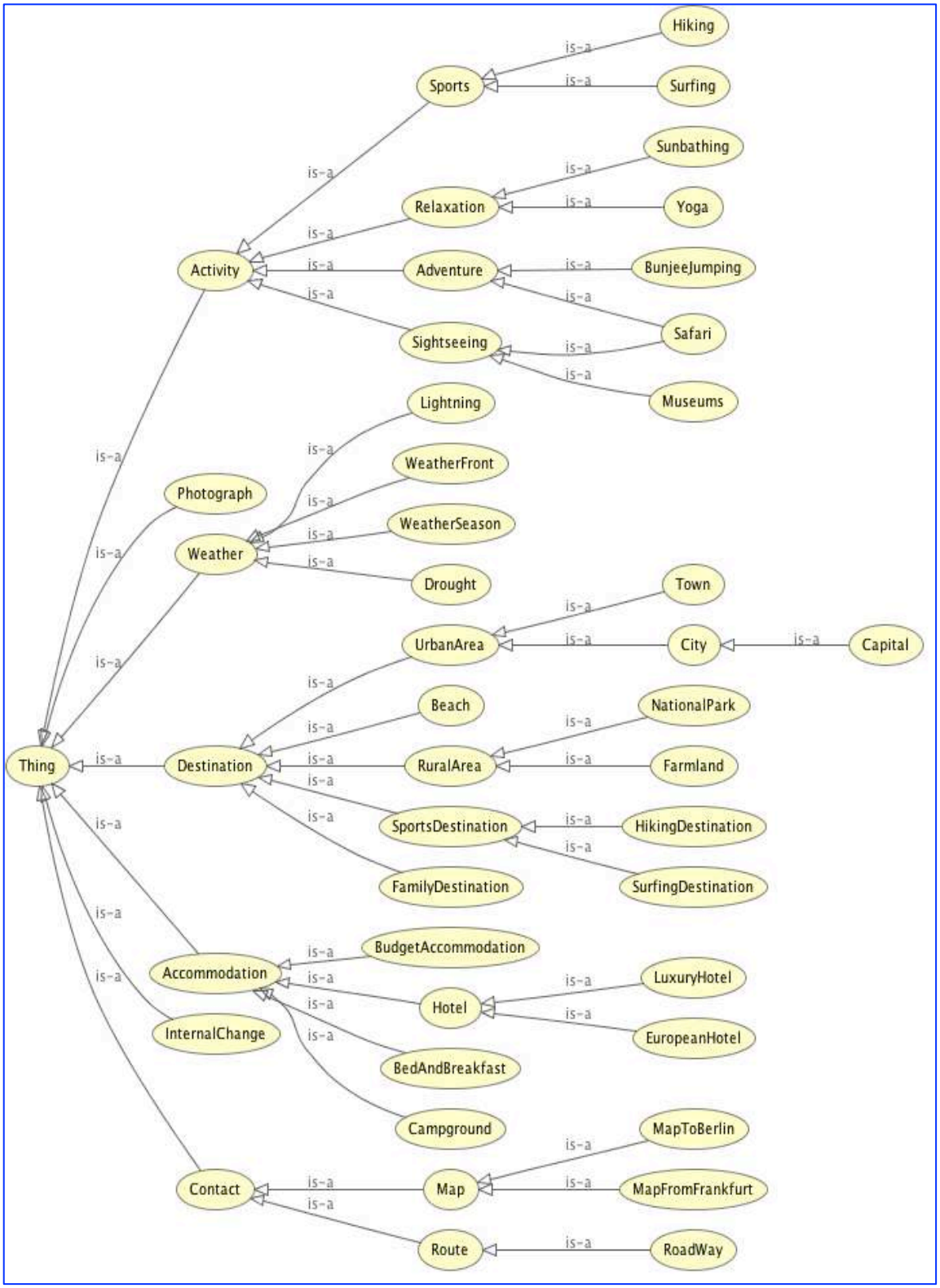


Figure 54. Extrait de l'ontologie des cibles $OntoT$ dans le domaine du voyage

Ainsi, l'évaluation du niveau de correspondance sémantique entre les cibles de l'intention de l'utilisateur ($\mathcal{T}_u$) et de l'intention du service ($\mathcal{T}_{svi}$) vérifie si un concept fourni par un service peut répondre à la cible demandée, *i.e.* s'il existe un lien entre les deux cibles dans l'ontologie $OntoT$. Nous évaluons ce niveau de correspondance en nous basant sur les travaux de

(Paolucci et al., 2002), présentés dans la section 3.5.2.1.1, qui mettent en relation un service demandé par rapport à un ensemble de services offerts. Dans les travaux de Paolucci et al. (Paolucci et al., 2002), l'analyse des correspondances repose sur la détermination de la relation de subsomption (*plug-in* et *subsume*) entre deux concepts. Cette relation de subsomption est comparable à la notion d'héritage puisqu'elle permet de déterminer si un concept est plus spécifique ou plus générique qu'un autre concept. Cette relation est intéressante dans le cadre de notre mise en correspondance des cibles, puisqu'elle permet de déterminer à quel point les deux cibles sont similaires, en se basant sur les liens de généralisation ou de spécialisation qui les relient dans l'ontologie $OntoT$. Dans ces travaux, ces auteurs suivent quatre niveaux distincts, présentant les relations hiérarchiques possibles entre deux concepts dans une ontologie :

- ***Exact*** : le concept demandé correspond exactement au concept proposé ;
- ***Plug-In*** : le concept demandé est compris dans le concept proposé ;
- ***Subsume*** **:** le concept demandé comprend (*subsume*) le concept proposé ;
- ***Fail*** : les deux concepts ne sont pas liés.

Nous présentons, plus en détails dans la Figure 55, notre algorithme de mise en correspondance des cibles.

Procedure Target Matching (T_u, T_{svi}, $OntoT$)

(1) T_{score} = 0 /* *Degré de similarité entre la cible de* I_u (T_u) *et la cible de* I_{svi} (T_{vi}) */
(2) MatchTargetType = null /* *la relation entre* T_u *et* T_{vi} *dans l'ontologie* $OntoT$*/
(3) link = *0* /* *le nombre de liens qui séparent* T_u *et* T_{vi} *dans l'ontologie* $OntoT$*/
(4) MatchTargetType = *GetMatchTargetType* (T_u, T_{svi}, $OntoT$)
(5) **If** MatchTargetType == « subsume » || MatchTargetType == « plug-in » **Then**
(6) link = *GetLink* (T_u, T_{svi}, $OntoT$)
(7) T_{score} = 1 / (link+1)
(8) **Else If** MatchTargetType == « exact » **Then**
(9) T_{score} = 1
(10) **Else**
(11) T_{score} = 0
(12) **End If**
(13) **Return** T_{score}
(14) **End Procedure**

Figure 55. Algorithme de la mise en correspondance des cibles

L'algorithme de mise en correspondance des cibles prend en entrée une *cible de l'intention de l'utilisateur* (T_u), une *cible de l'intention du service* (T_{svi}), ainsi qu'une *ontologie de cibles* ($OntoT$). Il produit à la sortie le *degré de similarité* (T_{score}) entre T_u et T_{svi}.

Cet algorithme commence par déterminer la relation qui existe entre $\mathcal{T}_u$ et $\mathcal{T}_{svi}$ dans l'ontologie *OntoT* (ligne 4) selon les quatre niveaux déterminés ci-dessus. Si $\mathcal{T}_u$ représente le même concept que $\mathcal{T}_{svi}$ dans *OntoT*, alors le *MatchTargetType* aura comme résultat un « *exact* ». Sinon, si $\mathcal{T}_u$ subsume $\mathcal{T}_{svi}$, alors *MatchTargetType* aura comme résultat un « *subsume* ». Sinon, si $\mathcal{T}_{svi}$ subsume $\mathcal{T}_u$, alors *MatchTargetType* aura comme résultat un « *plug-in* ». Sinon, si aucune relation entre $\mathcal{T}_u$ et $\mathcal{T}_{svi}$ n'est retrouvé ou bien si $\mathcal{T}_u$ et/ou $\mathcal{T}_{svi}$ n'existent pas dans l'ontologie *OntoT*, alors le *MatchTargetType* sera un « *fail* ».

Ensuite, si les deux cibles $\mathcal{T}_u$ et $\mathcal{T}_{svi}$ sont liées par une relation « *subsume* » ou « *plugin* » (ligne 5), alors à ce stade on se base sur la mesure de similarité sémantique pour calculer la distance qui les sépare. Ceci consiste à calculer d'abord le nombre de liens qui les séparent dans *OntoT* (ligne 6). Par la suite, le degré de similarité ($\mathcal{T}_{score}$) est calculé selon la Formule 3 (ligne 7). Sinon, si les deux cibles sont les mêmes (ligne 8), alors le degré de similarité ($\mathcal{T}_{score}$) aura 1 (ligne 9). Dans le cas contraire, $\mathcal{T}_{score}$ sera retourné avec un score égal à zéro (ligne 11).

L'algorithme se termine et retourne comme résultat final le score du degré de similarité entre les deux cibles ($\mathcal{T}_{score}$) (ligne 13).

7.2.2.3.2. Mise en correspondance des verbes

La mise en correspondance des verbes *(Verb Matching)* représente la deuxième étape de la mise en correspondance intentionnelle. Cette étape considère la mise en correspondance sémantique entre les verbes, selon une ontologie de verbe (*OntoV*) (*cf.* section 6.3.1.3). Dans cette ontologie, un verbe peut avoir une relation avec les autres verbes décrits dans l'ontologie. Chaque relation relie un verbe à d'autres verbes voisins dont la signification est soit plus générale (*hyperonyme*), soit plus spécifique (*hyponyme*), mais aussi à des verbes similaires *(synonyme)*. Les deux relations « *hyperonyme* » et « *hyponyme* » ne sont autre que des relations de subsomption entre les verbes, comme on l'a décrit dans la section 7.2.2.3.1. Ces deux relations représentent le lien hiérarchique qui peut exister entre deux verbes dans l'ontologie *OntoV*. Cette ontologie *OntoV* décrit également la relation *« synonyme »* qui peut exister entre deux verbes décrit dans l'ontologie. Cette ontologie de verbes, comme l'illustre la Figure 56, est décrite dans le domaine du *voyage* utilisé dans l'évaluation du processus de découverte de services. Prenons, par exemple, le verbe *« book »,* il est synonyme avec le verbe *« reserve »*. Ce verbe a des relations d'hyponymie avec le verbe *« hold-open »* et d'hyperonymie avec le verbe *« schedule ».*

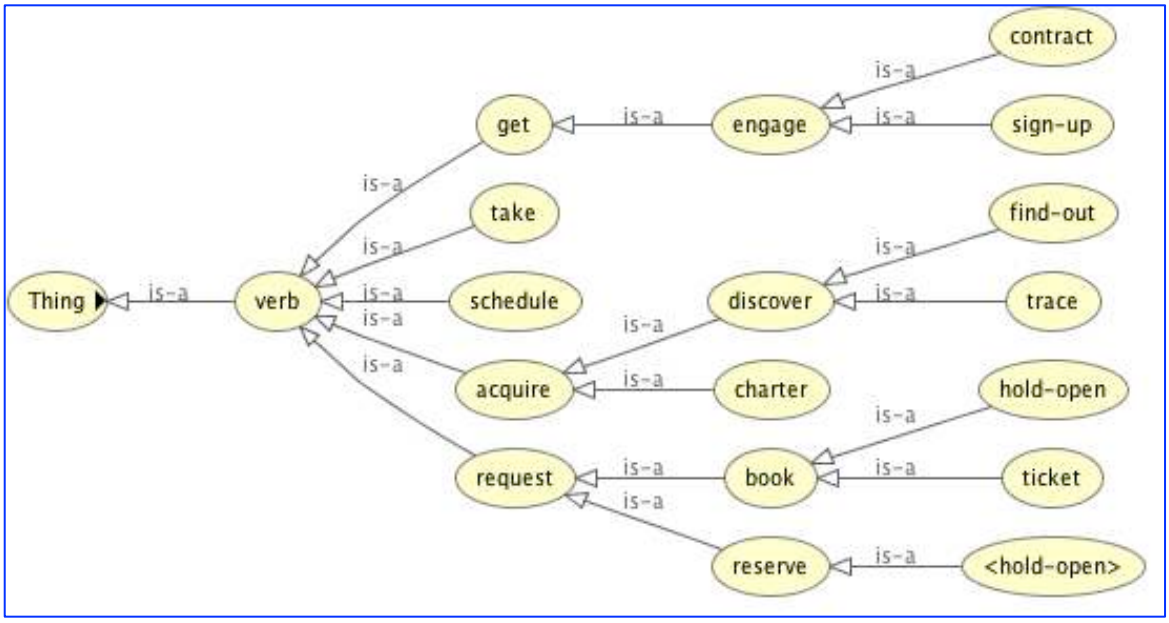


Figure 56. Extrait de l'ontologie de verbes $OntoV$

Dans les approches intentionnelles, il y a un risque d'ambiguïté (*e.g.* un même verbe qui a différents sens selon le domaine dans lequel il a été employé) et de complexité d'expression de l'intention. Ce problème relève du domaine de l'ingénierie de la requête, et a fait l'objet de multiples recherches, telles que les travaux de Gomez et al. dans leur approche GODO (Gomez et al., 2006) et Aljoumaa dans son approche PASiS (Aljoumaa, 2011) (*cf.* section 3.5.2.3.2). Dans le cadre de cette thèse, nous levons ce problème par l'usage d'un vocabulaire de domaine restreint matérialisé par des ontologies de domaines. Ainsi, nous ne prenons pas en compte le sens du verbe dans cette mise en correspondance.

L'évaluation de la mise en correspondance entre les verbes se fait grâce aux cinq niveaux suivants :

- ***Exact*** : le verbe demandé est équivalent au verbe proposé ;
- ***Synonym*** : le verbe demandé partage une signification commune avec le verbe proposé ;
- ***Hyponym*** : il y a une relation de subordination entre le verbe demandé et un verbe proposé, qui a une signification plus spécifique ;
- ***Hypernym*** : il y a une relation de subordination entre le verbe demandé et un verbe proposé, qui a une signification plus générale ;
- ***Fail*** : aucune relation entre les verbes ne peut être établie.

Nous présentons, plus en détails dans la Figure 57., notre algorithme de mise en correspondance des verbes lancé sur le verbe de l'intention de l'utilisateur ($\mathcal{V}_u$) et celui de l'intention du service ($\mathcal{V}_{svi}$). Dans cet algorithme, nous commençons par chercher le type de relation qui lie les deux verbes, selon les cinq niveaux cités ci-dessus. Par la suite, nous

calculons le score de cette mise en correspondance en suivant la Formule 3 de la mesure de similarité sémantique (*cf.* section 7.2.2.1).

Procedure Verb Matching ($\mathcal{V}_u$, $\mathcal{V}_{svi}$, $Onto\mathcal{V}$)
(1) $\mathcal{V}_{score}$ = 0 /* Degré de similarité entre le verbe de I_u ($\mathcal{V}_u$) et le verbe de I_{svi} ($\mathcal{V}_{svi}$) */
(2) MatchVerbType = null /* la relation entre $\mathcal{V}_u$ et $\mathcal{V}_{vi}$ dans l'ontologie $Onto\mathcal{V}$*/
(3) link = *0* /* le nombre de liens qui séparent $\mathcal{V}_u$ et $\mathcal{V}_{vi}$ dans l'ontologie $Onto\mathcal{V}$*/

(4) MatchVerbType = *GetMatchVerbType* ($\mathcal{V}_u$, $\mathcal{V}_{svi}$, $Onto\mathcal{V}$) /* retourne le type de relation entre $\mathcal{V}_u$, et $\mathcal{V}_{svi}$ dans $Onto\mathcal{V}$ */
(5) **If** MatchVerbType == « hypernym » || MatchVerbType == « hyponym » **Then**
(6) link = *GetLink* ($\mathcal{V}_u$, $\mathcal{V}_{svi}$, $Onto\mathcal{V}$)
(7) $\mathcal{V}_{score}$ = 1 / (link+1)
(8) **Else If** MatchVerbType == « exact » **Then**
(9) $\mathcal{V}_{score}$ = 1
(10) **Else If** MatchVerbType == « synonym » **Then**
(11) $\mathcal{V}_{score}$ = 0,9
(12) **Else**
(13) $\mathcal{V}_{score}$ = 0
(14) **End If**

(15) **Return** $\mathcal{V}_{score}$
(16) **End Procedure**

Figure 57. Algorithme de la mise en correspondance des verbes

L'algorithme de mise en correspondance des verbes prend en entrée le *verbe* indiqué dans l'intention de l'utilisateur ($\mathcal{V}_u$), le *verbe de l'intention du service* ($\mathcal{V}_{svi}$) et l'*ontologie de verbes* ($Onto\mathcal{V}$). Il produit à la sortie le *degré de similarité* ($\mathcal{V}_{score}$) entre $\mathcal{V}_u$ et $\mathcal{V}_{svi}$

Cet algorithme commence par déterminer la relation qui existe entre $\mathcal{V}_u$ et $\mathcal{V}_{svi}$ dans l'ontologie $Onto\mathcal{V}$ (ligne 4) selon les cinq niveaux déterminés ci-dessus. Si $\mathcal{V}_u$ représente le même concept que $\mathcal{V}_{svi}$ dans $Onto\mathcal{V}$, alors le *MatchVerbType* entre $\mathcal{V}_u$ et $\mathcal{V}_{svi}$ aura comme résultat un « *exact* ». Par contre, si $\mathcal{V}_u$ et $\mathcal{V}_{svi}$ sont reliés dans l'ontologie par la relation « *aSynonyme* », alors *MatchVerbType* aura comme résultat un « *synonym* ». Sinon, si $\mathcal{V}_u$ subsume $\mathcal{V}_{svi}$, alors *MatchVerbType* aura comme résultat un « *hypernym* ». Sinon, si $\mathcal{T}_{svi}$ subsume $\mathcal{V}_u$, alors *MatchVerbType* aura comme résultat un « *hyponym* ». Sinon, si aucune relation entre $\mathcal{V}_u$ et $\mathcal{V}_{svi}$ n'est retrouvé dans $Onto\mathcal{V}$ ou bien si $\mathcal{V}_u$ n'existent pas dans l'ontologie $Onto\mathcal{V}$, alors le *MatchVerbType* aura comme résultat un « *fail* ».

Ensuite, si les deux cibles $\mathcal{V}_u$ et $\mathcal{V}_{svi}$ sont liées par une relation « *hyponym* » ou « *hypernym* » (ligne 5), alors à ce stade on se base sur la mesure de similarité sémantique pour calculer la distance qui les sépare. Ceci consiste à calculer d'abord le nombre de liens qui les

séparent dans $Onto\mathcal{V}$ (ligne 6). Par la suite, le degré de similarité ($\mathcal{V}_{score}$) est calculé selon la Formule 3 (ligne 7). Sinon, si les deux cibles sont les mêmes (ligne 8), alors le degré de similarité ($\mathcal{V}_{score}$) aura un score égal à 1 (ligne 9). Sinon, si les deux cibles sont des synonymes (ligne 10), alors le degré de similarité ($\mathcal{V}_{score}$) aura un score égal 0,9 (ligne 11). Dans le cas contraire, $\mathcal{V}_{score}$ sera retourné avec un score égal à zéro (ligne 13).

L'algorithme se termine et retourne comme résultat final le score du degré de similarité entre les deux verbes ($\mathcal{V}score$) (ligne 15).

Après avoir introduit la mise en correspondance intentionnelle, nous détaillons dans la section suivante la mise en correspondance contextuelle.

7.2.2.4. La mise en correspondance contextuelle des services

Dans cette section, nous continuons la présentation de notre algorithme de mise en correspondance de services guidée par l'intention et le contexte, en présentant la partie du traitement contextuel de la requête de l'utilisateur. Dès qu'un service est sélectionné comme répondant à l'intention de l'utilisateur, comme présenté dans la section 7.2.2.3, la mise en correspondance contextuelle est déclenchée.

7.2.2.4.1. L'algorithme de mise en correspondance contextuelle

La mise en correspondance contextuelle *(Context Matching)* représente la deuxième étape de l'algorithme de mise en correspondance des services guidée par l'intention et le contexte. Cette étape de l'algorithme consiste à déterminer un degré de similarité entre la description de contexte de l'utilisateur ($Cx_{\mathcal{U}}$) et la description de contexte de chaque service répondant aux intentions de l'utilisateur ($C\mathcal{XR}_{svi}$). Le contexte $Cx_{\mathcal{U}}$ représente le contexte courant de l'utilisateur, alors que le contexte $C\mathcal{XR}_{svi}$ représente les conditions contextuelles dans lesquelles le service est valide et exécutable, *i.e.* le contexte requis du service (*cf.* Chapitre 5).

La description de contexte requis du service ($C\mathcal{XR}_{svi}$) représente un ensemble non-vide de conditions contextuelles, *i.e.* $C\mathcal{XR}_{svi} = \{cx_{svi}\}$, alors que la description de contexte de l'utilisateur ($Cx_{\mathcal{U}}$) représente un ensemble non-vide d'observations, *i.e.* $Cx_{\mathcal{U}} = \{cx_{\mathcal{U}}\}$ (*cf.* Définition 4). Chacune de ces conditions contextuelles cx_{svi} et de ces observations $cx_{\mathcal{U}}$ est décrite, selon le modèle de contexte (*cf.* section 6.4.1), par une *entité* (à qui le contexte se réfère), un *élément de contexte* (qui caractérise la propriété de l'entité observée) et un ensemble de *valeurs observées* ou de *conditions sur les valeurs* pour chaque élément.

La mise en correspondance contextuelle entre le contexte courant de l'utilisateur ($Cx_{\mathcal{U}}$) et celui requis du service ($C\mathcal{XR}_{svi}$), comme l'illustre la Formule 5, est déterminée à partir de l'évaluation (*Context Condition Matching)* de la satisfaction de toutes les conditions contextuelles (cx_{svi}) par les observations du contexte de l'utilisateur ($cx_{\mathcal{U}}$).

$$ContextMatching\ (CX\mathcal{R}_{svi},\ Cx_{\mathcal{U}}) = \{\exists\ cx_{svi} \in CX\mathcal{R}_{svi}\ ,\ \exists\ cx_{\mathcal{U}} \in Cx_{\mathcal{U}},\ ContextConditionMatching\ (cx_{svi},\ cx_{\mathcal{U}})\ \}$$

Formule 5. La mise en correspondance des conditions contextuelles

L'évaluation de ces conditions de contexte suit trois étapes. Pour chaque condition cx_{svi}, nous commençons par calculer le degré de similarité entre son entité ($\mathcal{E}cx_{svi}$) et l'entité de $cx_{\mathcal{U}}$ ($\mathcal{E}cx_{\mathcal{U}}$). Ceci permet de déterminer si l'observation et la condition se rattachent au même *sujet observé*. Ensuite, si $\mathcal{E}cx_{svi}$ est $\mathcal{E}cx_{\mathcal{U}}$ correspondent sémantiquement alors nous passons au calcul du degré de similarité entre l'élément de contexte de cx_{svi} ($\mathcal{E}lcx_{svi}$) et l'élément de contexte de $cx_{\mathcal{U}}$ ($\mathcal{E}lcx_{\mathcal{U}}$). Cette étape, permet de déterminer si cx_{svi} porte sur une condition sur le même élément de contexte (ou similaire sémantiquement) que celui décrit dans $cx_{\mathcal{U}}$. Finalement, si $\mathcal{E}lcx_{svi}$ est $\mathcal{E}lcx_{\mathcal{U}}$ correspondent sémantiquement alors l'évaluation de la satisfaction de la condition cx_{svi} peut être lancée sur l'ensemble de valeurs de $cx_{\mathcal{U}}$.

Chaque condition contextuelle est exprimée par un ensemble d'opérateurs (*cf.* section 6.4.3). D'une part, ces opérateurs peuvent être des *opérateurs simples*, tels que l'*égalité*, la *différence*, l'*intervalle*, la *supériorité* et l'*infériorité*. Par exemple, si la condition contextuelle d'un service (cx_{svi}) indique que le débit du réseau doit être *supérieur* à 12500 bits/s et que la valeur de l'observation ($cx_{\mathcal{U}}$) est égale à 13250, alors on peut dire que l'observation $cx_{\mathcal{U}}$ satisfait la condition cx_{svi}. D'autre part, ces opérateurs peuvent être plus *complexes*, par exemple les *opérateurs sémantiques* qui nécessitent des opérations de transformation d'une représentation à une autre. Par exemple, une évaluation d'une condition qui porte sur une *localisation représentée sous forme de coordonnées GPS* avec une observation qui représente la *localisation sous forme d'adresse postale* nécessite de transformer la localisation en GPS en localisation sous forme d'adresse, ou vice-versa, pour pouvoir les évaluer. Cette évaluation retourne pour chaque condition contextuelle un score de satisfaction. Ce score est égal à 1 si la condition est satisfaite ou égal à 0 si la condition n'est pas satisfaite.

$$Cx_{score} = \sum_{i=1}^{n} (w * ContextConditionMatching\ (cx_{svi},\ cx_{\mathcal{U}}))$$

Formule 6. Mise en correspondance contextuelle

Après l'évaluation des différentes conditions contextuelles appartenant à $CX\mathcal{R}_{svi}$, le score Cx_{score} est calculé par la somme pondérée des scores de chaque condition contextuelle (cx_{svi}), comme l'illustre la Formule 6. Un poids (w) est associé à chaque élément de contexte de la condition cx_{svi}. Ce poids, décrit plus en détail dans la section 7.2.2.4.2, reflète l'importance de l'élément de contexte par rapport aux préférences de l'utilisateur. Il est défini par l'utilisateur dans l'intervalle [0,1].

Procedure Contextual Matching (CxR_{svi}, Cx_{U}, *OntoCx*)

(1) Cx_{score} = 0 /* *Degré de similarité entre* CxR_{svi} *et le contexte de l'utilisateur* Cx_{U} */

(2) **For each** $cx_{svi} \in CxR_{svi}$ **Do**

(3) cx_{score} = 0

(4) Ecx_{svi} = *getEntity* (cx_{svi})

(5) $Elcx_{svi}$ = *getContextElement* (cx_{svi})

(6) w = *getContextElementWeight* ($Elcx_{svi}$)

(7) **For each** $cx_{U} \in Cx_{U}$ **Do**

(8) Ecx_{U}= *getEntity* (cx_{U})

(9) Ent_{score}= *ContextMatch* (Ecx_{svi}, Ecx_{U}, *OntoCx*) /* Mise en correspondance entre cibles*/

(10) **If** $Ent_{score} > \beta$ **Then**

(11) $Elcx_{U}$= *getContextElement* (cx_{U})

(12) Elt_{score} = *ContextMatch* ($Elcx_{svi}$, $Elcx_{U}$, *OntoCx*) /* Mise en correspondance entre éléments de contexte*/

(13) **If** $Elt_{score} > \beta$ **Then**

(14) $Cdcx_{svi}$= *getCondition* (cx_{svi}) /*récupérer la condition de cx_{svi} */

(15) $Valcx_{U}$= *getObservedValue* (cx_{U}) /*Récupérer la valeur observée de cx_{U} */

(16) $Repcx_{svi}$= *getRepresentation* ($Cdcx_{svi}$) /*récupérer la représentation de $Cdcx_{svi}$*/

(17) $Repcx_{U}$= *getRepresentation* ($Valcx_{U}$) /*récupérer la représentation de $Valcx_{U}$*/

(18) **If** *Match* ($Repcx_{svi}$, $Repcx_{U}$) **Then**

/* *Evaluer la satisfaction entre condition et observation de contexte* */

(19) $cdcx_{score}$ = *EvaluateSatisfactionCondition* ($Cdcx_{svi}$, $Valcx_{U}$)

(20) cx_{score} = (Ent_{score} + w * Elt_{score} + $cdcx_{score}$) / 3

(21) Cx_{score} = Cx_{score} + cx_{score}

(22) *break* /* *condition évaluée* */

(23) **End If**

(24) **End If**

(25) **End If**

(26) **End For**

(27) **End For**

(28) Cx_{score} = Cx_{score} / i /* i représente le nombre de conditions dans CxR_{svi} */

(29) **Return** Cx_{score}

(30) **End procedure**

Figure 58. Algorithme de la mise en correspondance contextuelle

Plus en détails, l'algorithme de mise en correspondance contextuelle, illustré à la Figure 58, prend en entrée la *description du contexte courant de l'utilisateur* ($Cx_{\mathcal{U}}$), la *description du contexte requis du service* (CXR_{svi}) et l'*ontologie multi-niveaux de contexte* (*OntoCx*). Il produit comme sortie le *degré de correspondance* (Cx_{score}) des conditions contextuelles du service (CXR_{svi}) avec contexte courant de l'utilisateur ($Cx_{\mathcal{U}}$).

Cet algorithme permet de calculer le degré de satisfaction des conditions contextuelles d'un service (CXR_{svi}) par rapport au contexte courant de l'utilisateur ($Cx_{\mathcal{U}}$). Ainsi, pour chaque condition contextuelle (cx_{svi}), l'algorithme récupère l'entité Ecx_{svi} et l'élément de contexte $Elcx_{svi}$ de la condition cx_{svi}, ainsi que le poids w attribué à cet élément de contexte à partir du modèle de profil de contexte (*cf.* section 7.2.2.4.2) (ligne 4-6). Par la suite, il cherche l'observation $cx_{\mathcal{U}}$ qui peut satisfaire la condition cx_{svi}.

Pour chaque observation de l'utilisateur $cx_{\mathcal{U}}$, l'algorithme récupère l'entité observée $Ecx_{\mathcal{U}}$ (ligne 8). Par la suite, une mise en correspondance entre les *entités* Ecx_{svi} et $Ecx_{\mathcal{U}}$ est effectuée afin de déterminer si la condition contextuelle et l'observation se rapportent à la même entité de contexte (ligne 9). Le degré de similarité Ent_{score} est attribué à cette mise en correspondance. Si Ent_{score} est inférieur au seuil β, alors cx_{svi} et $cx_{\mathcal{U}}$ ne représentent pas la même entité. Dans ce cas, l'algorithme passe à la vérification de l'observation suivante de l'utilisateur. Par contre, si Ent_{score} est supérieur au seuil β, alors l'algorithme procède à la vérification de la correspondance entre les éléments de contexte de cx_{svi} et $cx_{\mathcal{U}}$. Il récupère l'élément de contexte $Elcx_{\mathcal{U}}$ correspondant à $cx_{\mathcal{U}}$ (ligne 11). Ensuite, une mise en correspondance entre les *éléments de contexte* $Elcx_{svi}$ et $Elcx_{\mathcal{U}}$ est effectuée afin de déterminer si cx_{svi} et $cx_{\mathcal{U}}$ représentent une description du même élément de contexte (ligne 12). Le degré de similarité Elt_{score} est attribué à cette mise en correspondance (ligne 12). Si le score Elt_{score} est inférieur au seuil β, alors cx_{svi} et $cx_{\mathcal{U}}$ ne représentent pas une description du même élément de contexte. Dans ce cas, l'algorithme passe à la vérification de l'observation suivante de l'utilisateur. Par contre, si Elt_{score} est supérieur au seuil β, alors l'algorithme procède à la vérification de la satisfaction de la condition cx_{svi} par les valeurs observées de $cx_{\mathcal{U}}$.

L'étape de vérification de la satisfaction de la condition cx_{svi} par les valeurs observées de $cx_{\mathcal{U}}$ commence par la récupération de la description de la condition $Cdcx_{svi}$ de Cx_{svi} (ligne 14) et des valeurs observées $Valcx_{\mathcal{U}}$ de $cx_{\mathcal{U}}$ (ligne 15). Ensuite, il faut vérifier si ces deux dernières sont représentées de la même manière pour pouvoir les évaluer. Pour cela, l'algorithme commence par récupérer la représentation $Repcx_{svi}$ de la condition $Cdcx_{svi}$ (ligne 16) et la représentation $Repcx_{\mathcal{U}}$ des valeurs observées $Valcx_{\mathcal{U}}$ (ligne 17). Ensuite, il vérifie si la description des valeurs observées $Valcx_{\mathcal{U}}$ a la même représentation que la description de la condition $Cdcx_{svi}$ (ligne 18).

Dans le cas où cette contrainte n'est pas vérifiée, alors il est impossible de vérifier $Cdcx_{svi}$ et $Valcx_{\mathcal{U}}$ parce qu'ils n'ont pas la même représentation. Ainsi, la condition contextuelle cx_{svi} ne peut pas être satisfaite par l'observation contextuelle $cx_{\mathcal{U}}$ de l'utilisateur. Dans le cas contraire,

l'algorithme procède à l'évaluation de la satisfaction de $Cdcx_{svi}$ par $Valcx_{\upsilon}$ (ligne 19). Cette évaluation consiste, d'une part, à récupérer l'*opérateur* de la condition $Cdcx_{svi}$ et ses valeurs, ainsi que la valeur de $Valcx_{\upsilon}$. Par la suite, il faut vérifier, selon cet opérateur, si la valeur $Valcx_{\upsilon}$ satisfait la condition $Cdcx_{svi}$. Ceci est traité par la fonction *EvaluateSatisfactionCondition* qui retourne un degré de satisfaction $cdcx_{score}$ entre $Cdcx_{svi}$, $Valcx_{\upsilon}$ (ligne 19). Suite à ce score, l'algorithme calcule le score final (cx_{score}) de satisfaction de la condition cx_{svi} par l'observation cx_{υ} (ligne 20).

Finalement, le score global (Cx_{score}) de satisfaction de toutes les conditions contextuelles de CXR_{svi} par toutes les observations de Cx_{υ} est incrémenté par cx_{score} (ligne 21). L'algorithme sort ensuite de la boucle et passe à la condition suivante de CXR_{svi} (ligne 22). Une fois toutes les conditions vérifiées, l'algorithme se termine et retourne comme résultat final le score de la mise en correspondance contextuelle (Cx_{score}) (ligne 29).

7.2.2.4.2. Modèle de profil de contexte

Les informations de contexte, présentées dans la section 6.4.1, n'ont pas toutes la même importance lors d'un processus de découverte. Par exemple, un technicien qui ne se déplace pas souvent peut considérer que l'observation de sa localisation est peu pertinente (parce que trop intrusive ou pas assez significative) alors qu'un commercial qui se déplace chez le client peut considérer la localisation comme discriminatrice dans la sélection d'un service. Leur importance peut, en effet, varier d'un utilisateur à un autre en fonction de ses préférences. Par conséquent, nous proposons d'associer aux éléments de contexte la notion de « *poids* » (w), illustrée dans l'algorithme de mise en correspondance contextuelle à la Figure 58.

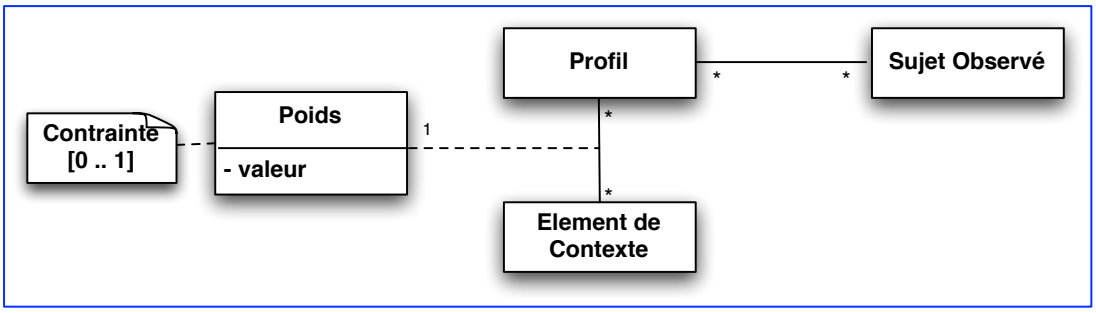


Figure 59. Modèle du profil de contexte

Notre objectif est de clarifier, à partir de cette notion de poids, l'importance d'un élément de contexte en fonction du domaine et des préférences de l'utilisateur. Pour cela, nous proposons le modèle du profil de contexte, présenté dans la Figure 59, lequel permet à chaque utilisateur d'attribuer des poids pour des éléments de contexte d'un sujet observé. Un *profil* représente les préférences d'un utilisateur concernant les informations de contexte. Ces préférences sont représentées comme un profil attribué à chaque *sujet observé*. Nous définissons un *poids* que le propriétaire du profil alloue à chaque élément de contexte. Ce

poids, dont la valeur est comprise entre 0 et 1, reflète l'importance de l'élément de contexte pour son propriétaire. Il est représenté en utilisant l'échelle indiquée ci-dessous :

- **Null** : 0.0
- **Pauvre** : [0.1 - 0.3]
- **Moyen** : [0.4 - 0.6]
- **Bon** : [0.7 - 0.9]
- **Excellent** : 1.0

L'importance d'un l'élément de contexte est ainsi proportionnelle à son poids. Lorsque le poids diminue, l'importance de l'élément de contexte diminue. Son impact sur le processus de découverte est amoindri. Inversement, lorsque le poids augmente, l'importance de l'attribut augmente. Il devient alors plus significatif dans le processus de découverte de services. Ce poids est utilisé lors de la mise en correspondance entre le contexte courant de l'utilisateur et les conditions contextuelles d'un service. En proposant ce modèle de profil de contexte, nous avons l'intention de promouvoir les éléments de contexte qui sont considérés comme les plus pertinents pour un utilisateur donné. En considérant ce profil, un élément de contexte ayant un poids plus faible (par exemple, qui n'est pas particulièrement intéressant pour l'utilisateur) sera moins influent lors du calcul du degré de la mise en correspondance contextuelle qu'un autre attribut avec un poids plus important. Même si cet élément de contexte participe au processus de mise en correspondance, le poids attribué à lui va diminuer son importance, et en conséquence le score de contexte calculé tiendra compte en priorité des éléments considérés comme prioritaires pour l'utilisateur.

Ce mécanisme permet à un utilisateur de définir, pour un sujet observé, un ensemble de profils représentant ses préférences. Grâce à cette notion de profil, il est possible d'améliorer la découverte et la sélection du service le plus approprié qui peut intéresser l'utilisateur.

7.2.2.5. La complexité de l'algorithme de découverte de services guidée par le contexte et l'intention

Afin de déterminer la complexité de notre algorithme de découverte de services guidé par l'intention et le contexte, nous avons reformulé notre algorithme selon la structure suivante :

Procédure Découverte de Services
Initialisation des variables Iscore, Cscore, Sscore
Début
Pour chaque *ensemble S1 (boucle 1)*
    Calculer la mise en correspondance intentionnelle (Iscore)
    Si *Iscore* > α
        Pour chaque *ensemble S2 (boucle 2)*
            Pour chaque *ensemble S3 (boucle 3)*
                Calculer la mise en correspondance contextuelle (Cscore)
            Fin pour

```
            Fin pour
            Si Cscore > β
                  Calculer le score final de mise en correspondance (Sscore)
            Fin Si
      Fin Si
Fin pour
Retourner le service le plus approprié
Fin
```

Trois ensembles sont considérés : (i) **S1** ⊂ {1, 2, 3, …, N} représente l'ensemble des services disponibles dans le répertoires de services ; (ii) **S2** ⊂ {1, 2, 3, …, M} représente l'ensemble des conditions contextuelles décrites dans le contexte requis du service ; et (iii) **S3** ⊂ {1, 2, 3, …, L} représente l'ensemble des observations contextuelles décrites dans le contexte courant de l'utilisateur. Dans le cadre de notre algorithme, nous considérons que N > M et N > L puisque le nombre de services maximal qui peut exister dans le répertoires de services dépasse le nombre d'observations et de conditions contextuelles qui peuvent être décrites.

Selon cette structure de l'algorithme, la *boucle externe 1* s'exécute N fois. Chaque fois que la boucle 1 s'exécute, la *boucle interne 2* s'exécute M fois. Ensuite, chaque fois la *boucle 2* s'exécute, la *boucle interne 3* s'exécute L fois. Par conséquent, les instructions de la *boucle 3* s'exécutent au total *N*M*L* fois. Ainsi, la complexité de notre algorithme est *O(N*M*L)*. Dans un cas particulier commun où la condition d'arrêt de la *boucle 2* est inférieure à N au lieu d'être inférieure à M et la condition d'arrêt de la *boucle 3* est également inférieure à N au lieu d'être inférieure à L (c'est-à-dire, les boucles internes s'exécutent également N fois), la complexité totale pour les trois boucles est $O(N^3)$, une complexité polynomiale de degré trois.

7.3. Implementation et evaluation

7.3.1. Les Technologies utilisées

Dans le cadre de cette thèse, nous présentons une première version de l'implémentation et l'évaluation de notre processus de découverte de services selon l'intention et le contexte. Pour cela, nous nous sommes basées sur un ensemble de technologies, à savoir :

- ***Jena – Framework du Web Sémantique pour Java*** (Carroll et al., 2004) représente un *framework* open source Java pour la création d'applications du Web sémantique. Il fournit un environnement de programmation pour RDF, RDFS et OWL, SPARQL et comprend un moteur d'inférence à base de règles. Le *framework* Jena comprend, entre autre, une API RDF pour la lecture, le traitement et l'écriture RDF en XML, une API d'ontologie pour la manipulation des ontologies OWL et RDFS, un moteur de requête SPARQL. Ce *framework* est utilisé principalement dans la partie persistance du projet.

Il permet la lecture, l'écriture et la vérification des ontologies utilisées dans le processus de découverte ;

- ***OWL-S API*** (OWL-S API, 2012) fournit une API Java pour l'accès à la lecture, l'écrire et l'exécution des descriptions de services OWL-S. Les structures de données de l'API ont été conçues en étroite collaboration pour correspondre à la définition de l'ontologie OWL-S. Dans notre cas, cette API sera utilisée afin de manipuler les descriptions de services, telles que l'obtention des intentions et des contextes de chaque service pour le calcul du score de la mise en correspondance.
- ***Pellet – Raisonneur OWL pour Java*** (Parsia et Sirin, 2004) fournit des services de raisonnement complet pour les ontologies OWL. Il présente des performances qui sont acceptables, voire très bonnes et un vaste *framework*. Il est écrit en Java et est *open source*. Dans le cadre de notre processus de découverte de services, il est utilisé à la fois dans l'API Jena et OWL-SIC, pour faire le raisonnement sur les ontologies et les descriptions de services ;
- ***OWLS-TC – Jeux de test pour la récupération des services OWL-S*** (Fries et al., 2005) destiné à soutenir l'évaluation de la performance des algorithmes de mise en correspondance des services OWL-S. La version 2 de OWLS-TC offre 576 services Web sémantiques écrits en OWL-S 1.0 et OWL-S 1.1 à partir de 7 domaines, à savoir : éducation, soins médicaux, nourriture, voyage, communication, économie et armé. Dans le cadre de ces expérimentations, notre choix de travailler sur le domaine du *voyage* pour illustrer les performances et la qualité de notre processus de découverte, vient en partie de la disponibilité de ce jeu de test.

Dans la section suivante, nous présentons notre implémentation du processus de découverte de services guidée par l'intention et le contexte.

7.3.2. Implémentation du processus de découverte de services

7.3.2.1. Implémentation du descripteur de services

Notre processus de découverte de services se base sur une extension OWL-SIC que nous avons présentée dans le Chapitre 6. Cette extension comprend la description des intentions qu'un service est capable de satisfaire et une description du contexte requis du service ($\mathcal{CXR}$) et du contexte dans lequel s'exécute le service ($\mathcal{Cx}$). Ainsi, sur la base de l'API OWL-S (OWL-S API, 2012), nous avons développé l'API OWL-SIC en langage JAVA permettant de manipuler cette description de services en fonction de leurs intentions et contextes.

Par ailleurs, afin de valider cette mise en œuvre, le jeu de tests OWLS-TC 2 a été utilisé. Nous avons choisi cette collection de test, car elle fournit un grand nombre de services issus d'une large variété de domaines, des requêtes de test et des ontologies pertinentes. Nous avons enrichi les descriptions de services proposés par OWL-S-TC 2 afin d'inclure à ces services une description intentionnelle et contextuelle. Ces descriptions de services sont, par la suite, utilisées pour évaluer le processus de découverte de services selon l'intention et le contexte.

7.3.2.2. Implémentation du processus de découverte de services

Le processus de découverte de services, que nous présentons dans le cadre de ce chapitre, a été implémenté en langage Java en se basant sur la plateforme de services proposée dans (Schulthess, 2011). Cette implémentation s'organise autour d'un certains nombre d'interfaces Java. Ces interfaces sont utilisées dans la perspective d'assurer un caractère flexible, réutilisable et échangeable de notre architecture. Nous cachons l'implémentation de l'algorithme de découverte de services derrière ces interfaces. Les implémentations de celles-ci se basent sur le patron de conception « *strategy pattern* » pour fournir un changement souple de stratégie, en se basant un fichier de configuration (*config.properties*).

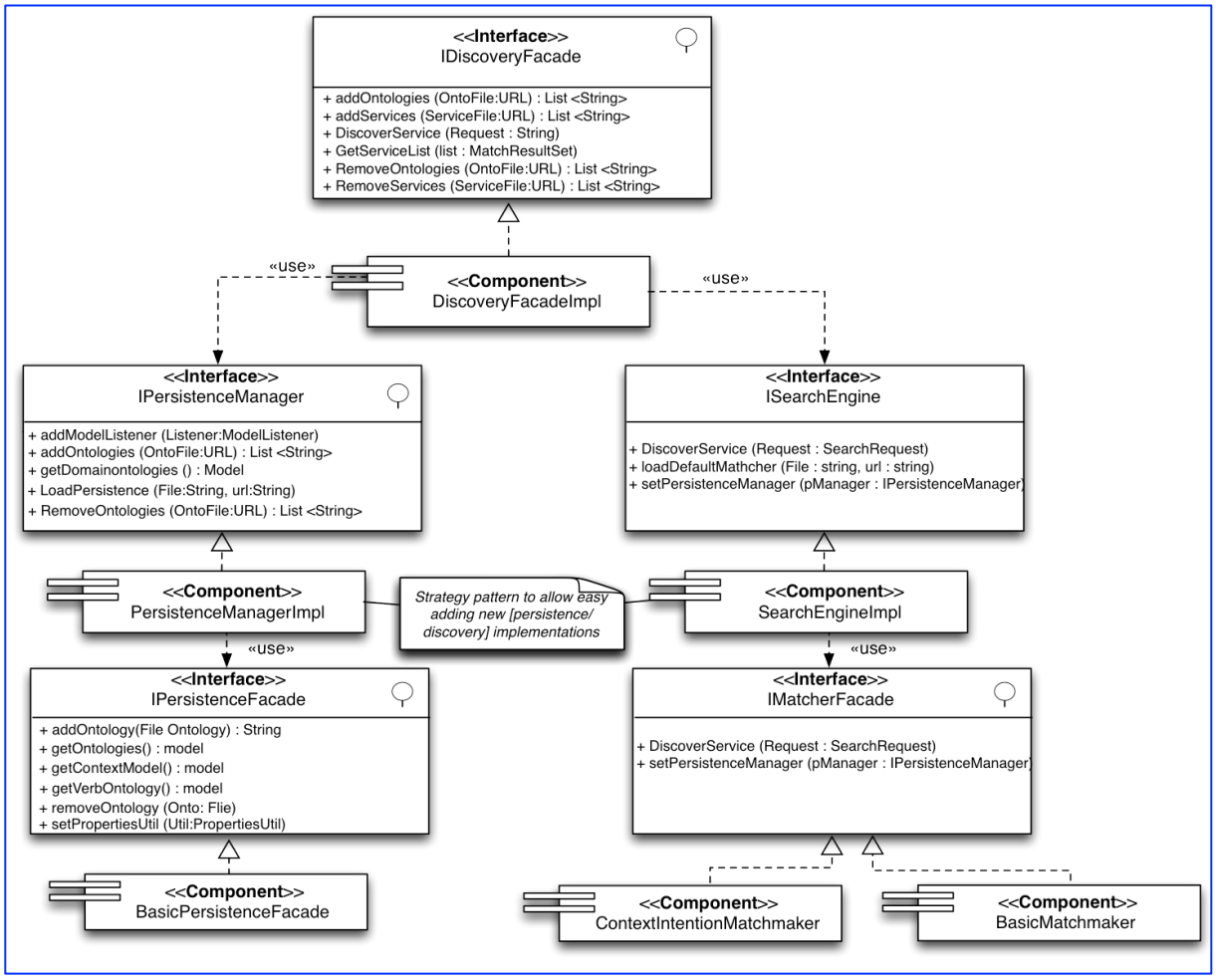


Figure 60. Éléments du mécanisme de découverte sémantique de services (d'après Schulthess (Schulthess, 2011))

Ainsi, notre implémentation du processus de découverte est organisée, essentiellement, autour de trois interfaces de bases, comme l'illustre la Figure 60, à savoir :

- L'interface « ***IDiscoveryFacade »*** représente le point d'entrée de l'application et offre un ensemble de méthodes qui supportent la gestion des ontologies, la découverte de services et leur sélection, telles que *addontologies*, *removeontologies, addServices, removeServices* et *DiscoverService* ;

- L'interface « ***IPersistenceManager »*** agit comme une façade entre la *« ServiceManager »* et le répertoire de services et d'ontologies, ce qui permet l'accès et le chargement des descriptions des services et des ontologies ;
- L'interface « ***ISearchEngine »*** est responsable de la recherche du service le plus approprié à une requête donnée. Il utilise une interface *« IMatcherFacade »* qui agit comme une façade entre le *« SearchEngine »* et l'API des descriptions de service. Ce *« IMatcherFacade »* est utilisé afin de donner plus de flexibilité à notre implémentation. Il permet d'étendre facilement le mécanisme de découverte de services par l'ajout de nouveaux algorithmes de découverte.

Dans l'architecture, l'interface « *ISearchEngine* » fournit la méthode « *loadDefaultMatcher »* qui permet de déterminer la stratégie à mettre en place. Elle fournit également une méthode principale de découverte de services, appelée « *DiscoverServices »*. Cette méthode fait appel à la bonne méthode de découverte, selon la stratégie spécifiée au préalable dans le fichier de configuration, laquelle doit découvrir selon l'intention et le contexte courant de l'utilisateur, le service le plus approprié.

Le composant « *SearchEngineImpl* » implémente l'interface « *ISearchEngine »* laquelle fournit les méthodes de découverte de services qui peuvent être utilisées par l'application. Chaque méthode doit être capable de proposer une liste de services ordonnée par leurs scores de mise en correspondance. A partir de ces scores, la méthode de sélection des services utilisée sélectionne le service qui est le plus approprié à l'intention de l'utilisateur dans son contexte courant. Afin de permettre l'évolution de l'API, cette première implémentation « *SearchEngineImpl* » utilise l'interface *« IMatcherFacade »,* laquelle offre une interface générale pour l'implémentation de nouvelles méthodes de découverte de services, appelées « *matchmaker* ». La sélection du « *matchmaker »* à déployer se fait ainsi par la simple édition d'un fichier de propriétés.

Dans nos expériences, nous proposons deux implémentations de « *matchmaker »,* correspondant à deux processus de découverte de services distincts :

- Le « ***BasicMatchmaker »*** utilise uniquement les références aux informations d'entrée (*input*) et de sortie (*output*) fournies par l'utilisateur pour procéder à la sélection des services. Cette implémentation agit comme une implémentation de référence dans nos évaluations, puisqu'elle adopte une approche traditionnelle basée sur les spécifications fonctionnelles d'un service. L'utilisateur est obligé, dans ce cas, de spécifier les entrées et les sorties qu'il attend du service recherché, au lieu de se concentrer uniquement sur l'intention qu'il souhaite satisfaire.
- Le « ***ContextIntentionMatchmaker »*** met en œuvre notre processus de découverte de services avec une prise en charge du contexte et de l'intention. Pour cela, le « *ContextIntentionMatchmaker »* utilise l'API OWL-SIC, le *framework* Jena (Carroll et al., 2004) et le moteur de raisonnement Pellet (Parsia et Sirin, 2004). Pour chaque service disponible dans le répertoire de services, le « *ContextIntentionMatchmaker »* permet de calculer un score de mise en correspondance avec l'intention et le contexte

de l'utilisateur, en se basant sur la description étendue du service. Cette classe fait appel à deux autres classes, à savoir « *ContextMatching* » et « *IntentionMatching* ». Le « *ContextMatching* » est responsable de déterminer le score de la mise en correspondance contextuelle (cf. section 7.2.2.4). Le « *IntentionMatching* » capable de calculer le score de la mise en correspondance intentionnelle (cf. section 7.2.2.3). La séparation entre ces deux éléments permet l'évaluation de chaque composant et l'analyse de leur impact sur le mécanisme de découverte.

La section suivante analyse les résultats des expérimentations qui ont été menées afin de comparer le processus de découverte de services guidée par l'intention et le contexte que nous proposons à une approche plus traditionnelle, représentée par le « *BasicMatchMaker* ».

7.3.3. Evaluation du processus de découverte de services

L'évaluation des différentes méthodes de découverte de services a été effectuée sur un répertoire sémantique contenant un ensemble étendu de descriptions de services, issu de la collection de jeux de test OWLS-TC 2. Parmi les domaines disponibles, nous avons choisi les services relevant du domaine du *voyage*. Ce domaine contient autour de 400 descriptions de services, qui ont été enrichies avec des informations contextuelles et intentionnelles.

Dans le cadre de nos expérimentations, nous avons commencé par déployer notre algorithme de *découverte de services* sur une machine de processeur *Intel Core i5 1,3 GHz* avec une mémoire de 4 Go. Ensuite, nous avons choisi d'évaluer notre algorithme sur différentes machines déployées sur la grille de calcul Grid'5000 [3] afin d'étudier ses performances sur différentes configurations de serveur. L'ensemble des mesures que nous avons réalisées a pour objectif d'évaluer la validité des algorithmes proposés et l'adéquation des descripteurs étendus de service qui ont été implémentés. Deux observations principales se dégagent de ces expériences :

- ***Le passage à l'échelle*** : analyse l'impact du nombre de services disponibles sur le temps d'exécution des algorithmes de découverte de services ;
- ***Qualité du résultat*** : évaluation de l'efficacité des différents algorithmes vis-à-vis de leur capacité à trouver un service adéquat au contexte et à l'intention de l'utilisateur.

La première de ces observations, le *passage à l'échelle*, nous permet d'analyser la faisabilité du mécanisme proposé. Celui-ci doit être capable de passer à l'échelle et son temps d'exécution doit rester limité pour ne pas gêner l'expérience de l'utilisateur, malgré un nombre grandissant de services analysés. La seconde observation, la *qualité*, considère deux métriques majeures dans le domaine des services : la *précision* et le *rappel*. Ces métriques nous permettent d'analyser si le processus de découverte de services que nous proposons atteint

[3] www.grid5000.fr

réellement son objectif, puisqu'il est censé fournir des résultats mieux adaptés à l'utilisateur par rapport à une approche traditionnelle.

Ensuite, afin d'évaluer le passage à l'échelle et les qualités des résultats, nous avons formulé sept requêtes différentes relatives au domaine du voyage et qui doivent retrouver les services correspondants dans le répertoire de services (*cf.* section 7.3.2.1). Ces requêtes sont caractérisées par l'intention de l'utilisateur et son contexte courant, comme indiqué dans le Tableau 2. Le contexte représente une description du contexte courant de l'utilisateur. Les valeurs de ce contexte ont été choisies afin de démontrer l'influence du changement de contexte sur le choix du service et sur la satisfaction de l'intention de l'utilisateur. Dans ce scénario, un utilisateur souhaite pratiquer un sport pendant ses vacances. Cet utilisateur est intéressé par le surf (*surfing*). Ainsi, il recherche des destinations où ces sports peuvent être pratiqués. Par la suite, il souhaite réserver une chambre dans un hôtel, un *lodge* ou un *Bed-and-Breakfast* pendant cette période.

Tableau 2. Des situations plus complexes de l'utilisateur : Intention dans un contexte donné

Requête	Intention	Contexte Courant
Req 1	- Reserve Five Star European Hotel	**Context User 1 :** -DateTime.Season = Winter, -Profile.Age = 21, -DateTime.Time = Morning, -Location.City = France -Resource.Screen = 16, -Device = Intel Core 2 duo -Resource.Memory = 2048, -Resource.Network = Ethernet
Req 2	- Book-up Lodge	**Context User 2 :** -DateTime.Season = Summer, -Profile.Age = 23 -DateTime.Time = Evening, -Location.City = Tanzania -Resource.Network = Wifi, -Device = Intel Core 2 duo -Resource.Memory = 1024, -Resource.Screen =15
Req 3	- Search BedAndBreakfast	**Context User 1**
Req 4	- Find Contact	**Context User 3** -DateTime.Season = Winter, -DateTime.Time = Night -Profile.Age = 16, -Profile.Expertise = Low -Profile.Role = Student, -Location.City = Mexique -Location.Country = USA, -Resource.Network = Wifi -Device = Iphone 4, -Resource.Memory = 16 -Resource.Screen = 3
Req 5	- Search Destination	**Context User 1**
Req 6	- Locate Surfing Destination	**Context User 4** -DateTime.Season = Summer, -Profile.Age = 24 -DateTime.Time = Morning, -Profile.Expertise = High -Profile.Role = Student, -Location.City = Hawai -Location.Country = USA, -Resource.Network = 3G -Device = Ipad 2, -Resource.Memory = 32 -Resource.Screen = 9
Req 7	Look for Surfing Course	**Context User 5** -DateTime.Season = Summer, -Location.City = Germany -DateTime.Time = Evening, -Profile.Expertise = High -Profile.Role = Student, -Device = Intel Core 2 duo

		-Profile.Age = 23, -Resource.Network = Ethernet, -Resource.Memory = 2048, -Resource.Screen = 16

Ces requêtes sont formulées de différentes manières. Nous avons décrit des situations où les éléments de l'intention de l'utilisateur ne sont pas décrits dans les ontologies d'intention alors qu'il existe dans le répertoire de services un ensemble de services capables de satisfaire cette intention dans le contexte courant de l'utilisateur (*Req2* et *Req7*). Par exemple, dans la requête *Req2*, le verbe et la cible de l'intention « *book-up lodge* » ne sont pas décrits dans les ontologies de verbes et de cibles. Toutefois, il existe un ensemble de services capable de satisfaire cette intention et notamment ceux qui répondent à l'intention « *reserve accommodation* » qui est similaire à l'intention de la *Req2*. Ainsi, une intention exprimée par l'utilisateur dont le verbe et/ou la cible ne sont pas décrits dans les ontologies ne peut pas être traitée même s'il existe des services qui peuvent répondre à cette intention dans le contexte courant de l'utilisateur.

De plus, nous avons décrit des situations où les éléments de l'intention de l'utilisateur sont décrits dans les ontologies de verbes et de cibles. Ces situations sont spécifiées afin de montrer que le seuil de paramétrage, défini au préalable par le concepteur du système dans l'algorithme de découverte de services, peut éliminer certains services même s'ils sont capables de satisfaire la situation de l'utilisateur (*Req1*, *Req5* et *Req6*). Par exemple, dans la requête (*Req1*), la cible *fiveStarEuropeanHotel* représente un *plug-in* de la cible *accommodation*. Dans l'ontologie de cibles, il existe quatre liens qui séparent ces deux cibles. Toutefois, puisque le seuil de paramétrage testé ne permet d'accepter que trois liens entre deux concepts dans l'ontologie, certains services qui répondent à cette intention ne seront pas sélectionnés. Ceci peut nuire à la qualité des résultats en éliminant des services qui sont appropriés à la situation de l'utilisateur. De même, pour la requête 6 (*Req6*), le verbe *locate* représente un *hyponyme* du verbe *acquire* mais qui se trouve au niveau du quatrième nœud dans l'ontologie de verbes. Ainsi, des services qui répondent aux intentions « *acquire destination* » et « *acquire surfingdestination* », par exemple, ne seront pas sélectionnés. Dans la requête 5 (*Req5*), nous avons choisi d'évaluer une cible (*destination)* permettant de tester notre algorithme sur un balayage en largeur et profondeur sur le nœud *destination* dans l'ontologie de cibles. Ceci permet d'évaluer l'acceptabilité de la performance de l'algorithme de découverte de services. De plus, les services qui permettent de satisfaire l'intention « *search destination* », mais qui contiennent des cibles qui sont situées au-delà du troisième nœud, ne seront pas sélectionnés.

Finalement, nous avons évalué dans les requêtes 3 et 4 (*Req3* et *Req4*) des intentions qui peuvent être satisfaites par un ensemble de services dans le contexte courant de l'utilisateur. Il existe, dans le répertoire de services, un ensemble de services qui répondent à ces deux intentions « *search bedAndBreakfast* » et « *find contact* » mais dans des contextes qui sont différents des contextes courants de l'utilisateur décrits dans les *Req3* et *Req4* (même description de contexte mais avec des valeurs différentes). Par exemple, la requête 4 présente un contexte courant de l'utilisateur de nature assez complexe. D'une part, la description de ce contexte courant contient plusieurs observations contextuelles décrivant différents éléments

de contexte. D'autre part, il existe, dans le répertoire de services, un ensemble de services qui répondent à l'intention de la *Req4*. Toutefois, certains de ces services sont décrits par des conditions contextuelles portées sur les mêmes éléments de contextes décrits dans le contexte courant de l'utilisateur mais avec des valeurs différentes. Ainsi, l'algorithme de découverte de services, va devoir comparer sémantiquement ses éléments de contexte, même si le service ne correspond pas sémantiquement. Ceci va impacter la performance de notre algorithme, puisqu'il va prendre un temps d'exécution plus élevé afin d'évaluer la mise en correspondance contextuelle de tous ces éléments.

En résumé, nous synthétisons les caractéristiques des requêtes de l'utilisateur dans le Tableau 3. La performance est déterminée par rapport au temps d'exécution moyen mis par l'algorithme de découverte de services pour découvrir le service le plus approprié allant de ++ (temps d'exécution faible) au -- (temps d'exécution plus élevé).

Tableau 3. Caractéristiques des requêtes de l'utilisateur

Requête	Concept(s) n'existe pas dans l'ontologie	Complexité concept dans l'ontologie	Complexité contexte (couplage faible)	Performance
Req 1		✓ (cible)	✓	--
Req 2	✓			++
Req 3				+
Req 4			✓	-
Req 5		✓ (cible)	✓	--
Req 6		✓ (verbe)	✓	--
Req 7	✓			++

7.3.4. Le passage à l'échelle (Performance sur plusieurs configurations)

Le *passage à l'échelle* est la capacité d'un système ou dispositif informatique à s'adapter au rythme de la demande (Bondi, 2000). L'analyse de cette propriété est un facteur essentiel lors du développement des systèmes répartis car cette propriété permet de déceler les limitations possibles d'un système lors d'une montée en charge. Dans notre cas, cette montée en charge peut comprendre autant le nombre de clients (utilisateurs) que le nombre d'éléments dans le répertoire de services. De manière plus générale, dans les mécanismes de découverte de services, le passage à l'échelle a un impact direct sur le temps d'évaluation des requêtes, ce qui peut gêner l'expérience de l'utilisateur si ce temps devient trop important.

Dans les expériences que nous avons menées, nous avons analysé le passage à l'échelle des mécanismes de découverte de services à travers le temps moyen de traitement, lorsqu'on varie le nombre de services disponibles dans le répertoire de services. Nous présentons dans les sections suivantes les résultats expérimentaux obtenus sur différentes machines.

7.3.4.1.1. Machine type Desktop

Le temps de traitement des requêtes a été mesuré en variant le nombre de services dans le répertoire de services entre 10 et 400 services, ce qui est indiqué dans la Figure 61 par l'axe des abscisses « taille du répertoire de services ». Les résultats démontrent que le temps de réponse de l'algorithme de découverte suit une *tendance polynomiale de degré 3*. Ceci nous permet d'affirmer que le processus de découverte de services implémenté a un niveau satisfaisant de passage à l'échelle, dans le cadre d'un SIP.

Plus en détails, la Figure 61 présente la comparaison entre les trois algorithmes de découverte de services utilisés dans cette expérimentation : (*i*) un mécanisme de découverte de services se basant uniquement sur les **entrées** et les **sorties** d'un service (implémenté par la classe *BasicMatchFacade*), représentant ainsi une vision purement fonctionnelle ; (*ii*) un mécanisme de découverte purement **intentionnel**, basé uniquement sur l'intention de l'utilisateur (implémenté par la classe *ContextIntentionMatchmaker*, dans laquelle la découverte guidée par le contexte est désactivée) ; et (*iii*) le mécanisme de découverte de services guidé par l'**intention et le contexte** implémenté par la classe *ContextIntentionMatchmaker*.

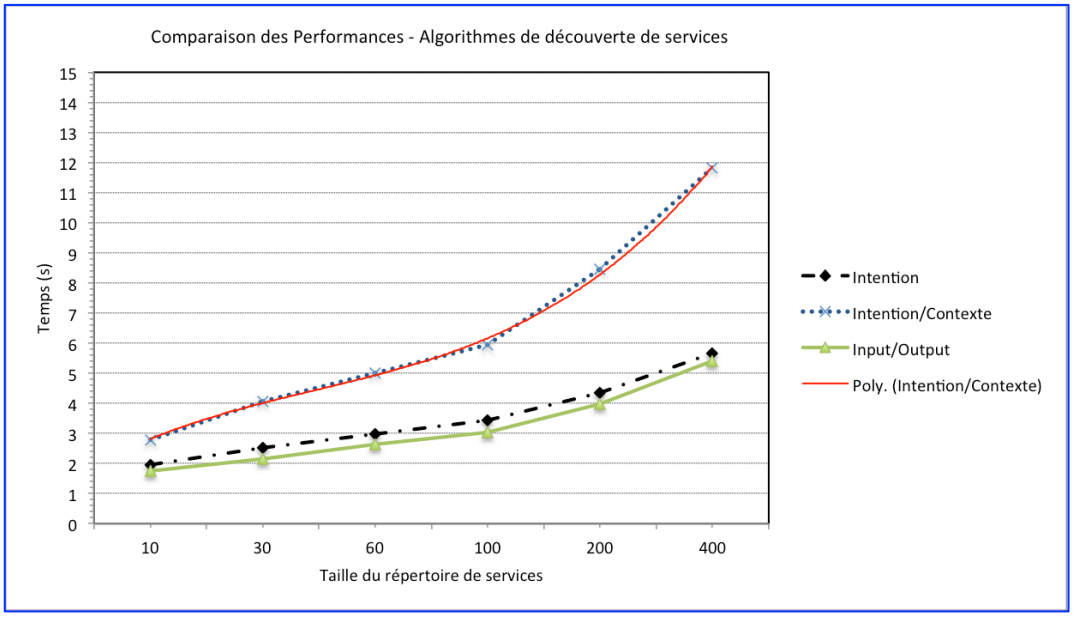


Figure 61.Comparaison des performances des mécanismes de découverte de services

Il convient d'observer que la variation purement contextuelle de la classe *ContextIntentionMatchmaker* n'a pas été considérée car l'intérêt de l'usage de contexte dans la découverte de services a déjà été démontré par de nombreux travaux, tels que (Ben Mokhtar et al., 2008), et a été largement discuté dans l'état de l'art (*cf.* Chapitre 2 et Chapitre 3).

Dans la Figure 61 nous observons que le mécanisme de découverte guidé par le contexte et l'intention (IC) a un temps moyen de réponse plus élevé que les autres algorithmes. Cette différence, surtout lorsqu'elle est comparée à l'algorithme d'entrée/sortie (IO), n'est pas

inquiétante et reste raisonnable. Nous pouvons remarquer que bien que nous ayons augmenté le nombre de services à traiter plus de quarante fois, le temps n'a augmenté que de quatre fois. Toutefois, la performance du mécanisme de découverte de services (IC) dépends du temps de traitement de l'intention de l'utilisateur et de son contexte courant. Par exemple, l'évaluation des situations de l'utilisateur décrites par les requêtes *Req2* et *Req7*, illustrées au Tableau 2, ne prend pas beaucoup de temps d'exécution comparées aux autres requêtes. En effet, dans la cas de l'évaluation des requête *Req2* et *Req7*, le mécanisme de découverte de services (IC) ne traite que la mise en correspondance intentionnelle et ne passe pas à la mise en correspondance contextuelle, puisque l'intention ne peut pas être satisfaite par l'ensemble des services. Tandis que, dans le cas de l'évaluation des autres requêtes, le mécanisme de découverte de services (IC) traite la mise en correspondance intentionnelle et la mise en correspondance contextuelle, puisque l'intention peut être satisfaite par un ensemble des services. Par exemple, l'évaluation des requêtes *Req4* et *Req5* ont mis beaucoup plus de temps d'exécution que les autres requêtes. Ceci, est dû à la richesse des éléments de l'intention dans les ontologies correspondances (plusieurs verbes et/ou cibles qui sont plus spécifiques et/ou plus génériques que le verbe et/ou la cible de l'intention), d'une part, et d'autre part, à la complexité de la description contextuelle.

De plus, cette implémentation représente une première version utilisée pour valider notre processus de découverte de services, que nous envisageons, à court terme, de l'optimiser en utilisant le traitement en parallèle des services à analyser, par exemple.

De ce fait, afin d'évaluer plus en détails les performances de notre algorithme de découverte de services, nous avons décidé de mener nos évaluations sur différentes configurations de machines. L'objectif ici est de vérifier le comportement de cet algorithme de découverte de services lors de son exécution sur des machines s'approchant d'avantage à ce qui serait utilisé dans un SIP.

7.3.4.1.2. Machine de type serveur

La plateforme Grid'5000 représente une grille de calcul expérimentale comportant différentes machines hétérogènes de type serveur (distribuées sur plusieurs institutions en France et à l'étranger), similaires à l'infrastructure du fournisseur de services. Cette plateforme représente un réseau atypique car il est composé de grappes de calcul hétérogènes, comme illustré au Tableau 4. Ainsi, le Grid'5000 présente le moyen d'accéder à différentes configurations techniques de type serveur (mis à disposition par l'Université de Reims Champagne-Ardenne).

Tableau 4. Grappes appartenant à Grid'5000

Cluster (année)	CPU	Processeur	Coeurs	RAM (Go)
Stremi (2011)	2	AMD 1.7 GHz	12	48
Graphene (2010)	1	Intel 2.53 GHz	4	16
Griffon (2009)	2	Intel 2.5 GHz	4	16

L'utilisation des ressources de Grid'5000 nous a permis ainsi de comparer la performance des algorithmes lorsqu'ils sont déployés sur des machines avec des caractéristiques différentes, comme illustré à la Figure 62.

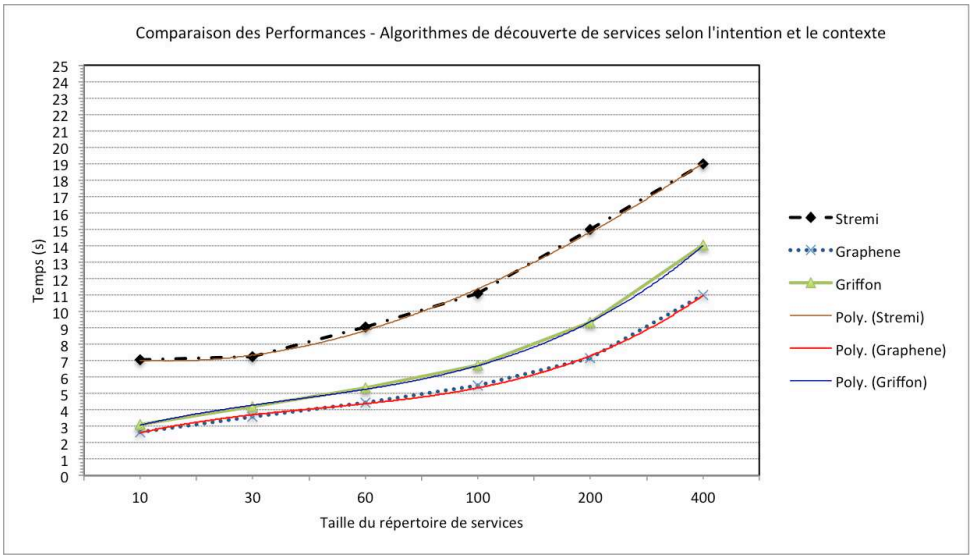


Figure 62. Performance du mécanisme de découverte de services selon l'Intention/Contexte sur le Grid'5000

De même que précédemment, le temps de traitement des requêtes a été mesuré en variant le nombre de services dans le répertoire de services entre 10 et 400 services. Les résultats illustrés à la Figure 62 démontrent que le temps de réponse de l'algorithme de découverte suit la même tendance polynomiale. Les variations les plus importantes sont observées lorsqu'on compare les différentes familles de processeurs (Intel versus AMD). Le nombre de cœurs n'a pas suffisamment influencé les résultats du fait que le prototype implémenté n'a pas été conçu pour le support aux architectures multi-cœur. Nous sommes certains que l'exploration parallèle du répertoire de services permettra de baisser considérablement le temps moyen de réponse, tout en gardant le passage à l'échelle observé dans cette expérience. Il faut noter que la Figure 62 présente seulement les courbes de performance pour l'algorithme Intention/Contexte pour des raisons de simplicité, car les observations ci-dessus sont aussi valables pour les autres algorithmes, comme le démontre la Figure 63.

La Figure 63 présente en détails la comparaison entre les trois algorithmes de découverte de services utilisés dans cette expérimentation. Ces trois algorithmes, appelés respectivement *IO* (*Input/Output*), *I* (*Intention*) et *IC* (*Intention/Contexte*), sont représentés par leurs performances sur la grappe Graphene. Cette grappe a été choisie par ses caractéristiques (nombre de cœurs, quantité de mémoire, etc.), qui sont les plus proches de celles qu'on trouve

sur le marché. Les autres grappes, à l'instar de *Stremi* par exemple, sont trop représentatives des machines pour le HPC[4] (quantité importante de mémoire, grand nombre de cœurs).

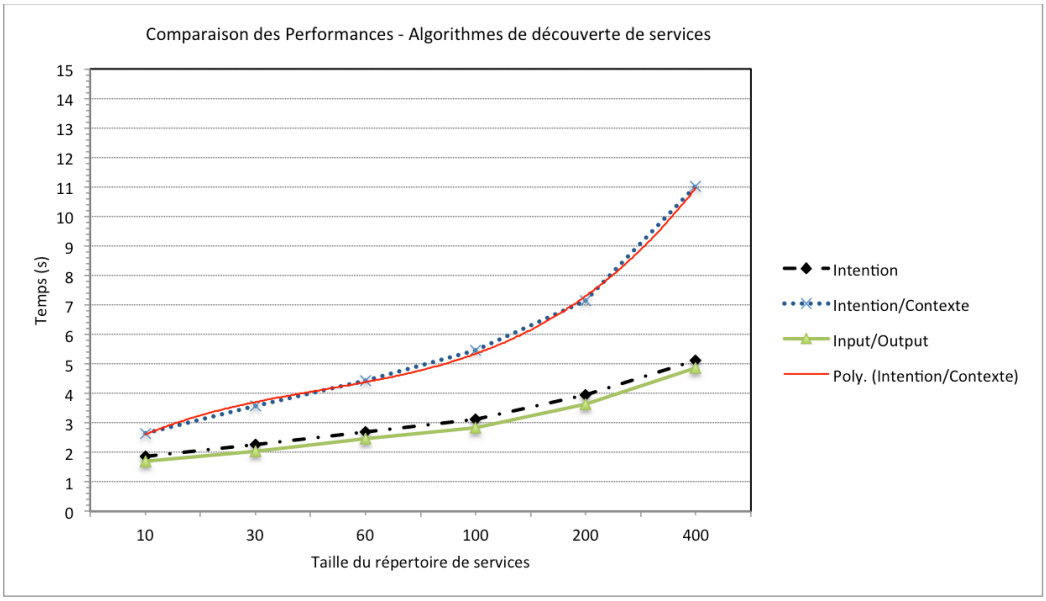


Figure 63. Comparaison des performances des mécanismes de découverte de services sur le cluster Graphene

Le résultat illustré à la Figure 63 démontre également que le mécanisme de découverte de services IC a un temps moyen de réponse plus élevé que les deux autres algorithmes. Toutefois, ce temps d'exécution obtenu demeure raisonnable pour une première version d'implémentation qui n'est pas optimisée. Ceci nous encourage à utiliser cette grille de performance comme une perspective d'optimisation de l'algorithme de découverte de services selon l'intention et le contexte en lançant le traitement en parallèle sur des sous ensembles de services répartis sur les machines distribuées dans une grappe de calcul.

Ceci est une donnée importante pour nous, car elle démontre la faisabilité de notre proposition sur des machines habituellement utilisées comme répertoire de services et ceci malgré une implémentation non optimisée pour les architectures multi-cœur. De plus, il ne faut pas oublier que l'algorithme IO utilise un mécanisme très simplifié lequel a un coût réduit mais, comme le démontre la prochaine section, il présente une qualité de résultat bien inférieure à celle de l'algorithme *Intention/Contexte* (IC).

7.3.4.2. Qualité des Résultats

Afin d'évaluer la qualité des résultats, nous avons considéré les deux métriques de qualité dans la sélection de services : la *précision* et le *rappel*. Ces deux métriques sont définies par deux ensembles, l'ensemble des *retrieved items* et l'ensemble des *relevant items*. La métrique *précision* indique la capacité d'un système à retrouver uniquement les *relevant items* (*i.e.* sans

[4] Calcul de haute performance (en anglais, *High Performance Computing*)

faux-positifs), alors que la métrique *rappel* mesure la capacité d'un système à retrouver tous les *relevant services* (Xiao et al., 2010). Ces deux métriques sont définies selon la Formule 7.

$$Précision = \frac{|\{relevant\ items\} \cap \{retrieved\ items\}|}{\{retrieved\ items\}}, Rappel = \frac{|\{relevant\ items\} \cap \{retrieved\ items\}|}{\{relevant\ items\}}$$

Formule 7. Calcul des métriques de précision et de rappel

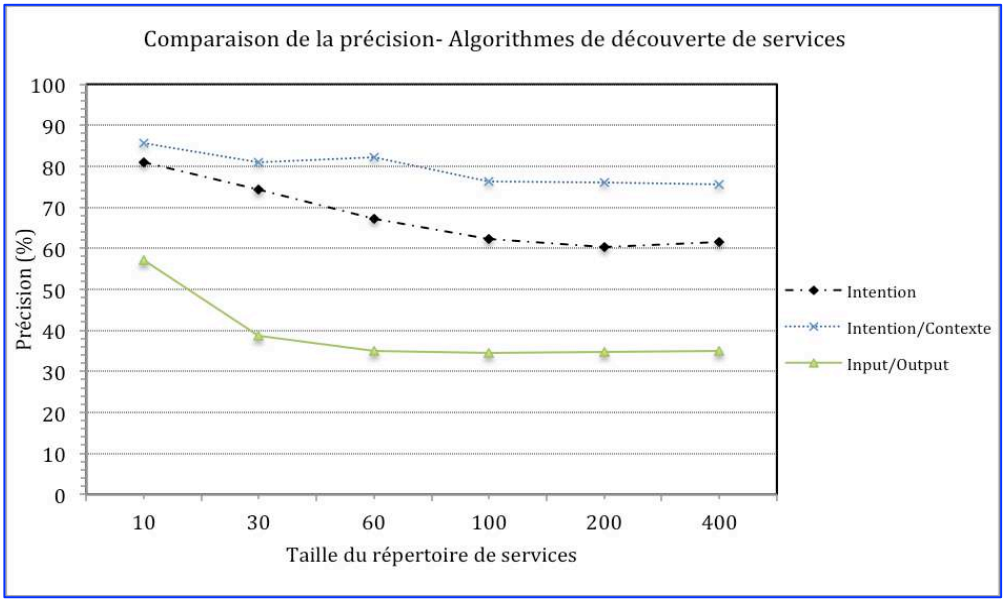


Figure 64. Comparaison de la métrique *Précision*

À partir des résultats obtenus (Figure 64 et Figure 65), nous pouvons observer que l'algorithme (IC) affiche, d'une manière générale, de bons résultats autant dans la métrique précision que dans celle de rappel. Les résultats présentés en Figure 64 indiquent que l'algorithme de découverte de services guidée par l'intention et le contexte (IC) présente un niveau de **précision** très supérieur à celui obtenu par l'algorithme de base (IO). En effet, la précision moyenne de l'algorithme IC est proche à 80%, alors que l'algorithme IO a une précision moyenne autour de 40%. Ces résultats indiquent que le mécanisme de découverte de services guidé par l'intention et le contexte a une plus grande chance de retrouver le service le plus approprié au contexte et à l'intention de l'utilisateur.

Les bons résultats obtenus pour la *précision* sont accompagnés par des résultats moins intéressants concernant le *rappel*, comme l'illustre la Figure 65. Effectivement, on observe que le taux de *rappel* moyen pour l'algorithme IC n'avoisine que les 67%. L'algorithme IO a aussi un taux moyen de 68%, mais ces résultats sont circonstanciels vu que cet algorithme n'est pas en mesure de choisir un service qui s'adapte au contexte de l'utilisateur ni à son intention. En effet, l'algorithme IO se limite à retourner l'ensemble de services qui peuvent concerner la requête, avec un risque important de faux-positifs (indiqué par sa *précision*).

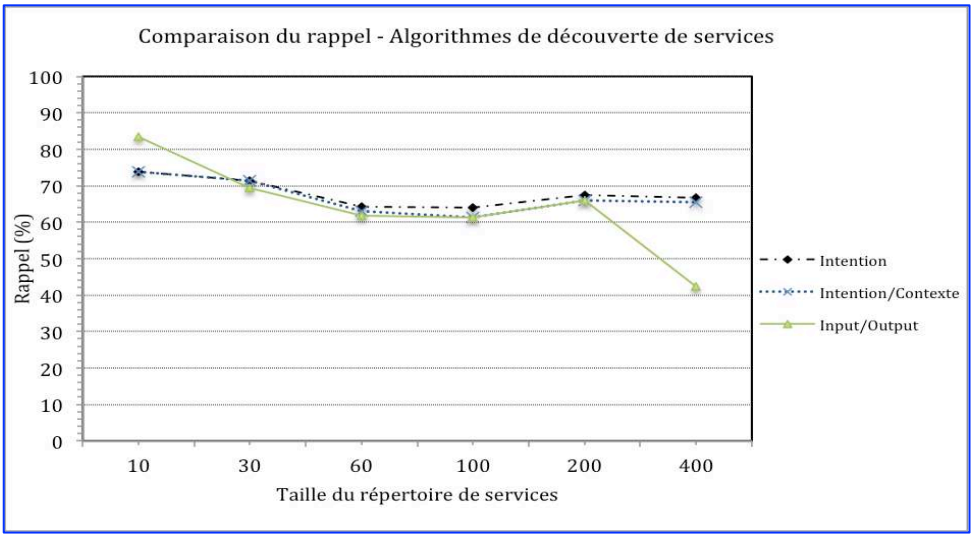


Figure 65. Comparaison de la métrique *Rappel*

Ces résultats peuvent être expliqués par l'évaluation de certaines situations qui peuvent nuire aux qualités des résultats obtenus. Par exemple, nous avons décrit des situations (*Req2* et *Req7* illustrées au Tableau 2) où les éléments de l'intention ne sont pas décrits dans les ontologies, alors qu'il existe dans le répertoire de services un ensemble de services capables de satisfaire cette intention dans le contexte courant de l'utilisateur. Dans ce cas, notre algorithme ne retourne aucun services. Ceci contribue à la dégradation de la qualité des résultats obtenant ainsi un *rappel* de 0%.

De plus, dans le cas des situations qui sont décrites par des intentions dont le verbe et/ou la cible sont assez génériques ou spécifiques (*Req1*, *Req5* et *Req6* décrites dans le Tableau 2), nous avons obtenus dans certains cas des rappels qui sont au dessous des 40%. Ainsi, lorsque le concepteur du système fixe un seuil de paramétrage très élevé dans l'algorithme de découverte de services, certains services qui peuvent répondre à l'intention de l'utilisateur dans son contexte courant ne seront pas sélectionnés, et ceci contribue à la dégradation de la qualité des résultats en terme de rappel.

Ainsi, contrairement à ce qui est observé avec l'algorithme IO, l'analyse conjointe des métriques *précision* et *rappel* démontre que l'algorithme IC réussit à trouver tous ou presque tous les services qui correspondent à l'intention et au contexte de l'utilisateur, et cela avec un faible taux de faux-positifs. Toutefois, ceci n'est valide que dans le cas d'une description complète et riche des ontologies et d'un bon paramétrage des seuils de mise en correspondance dans l'algorithme de découverte de services. Les seuils de mise en correspondance sont évalués au préalable par le concepteur du système d'une manière empirique, comme nous l'avons expliqué dans la section 7.2.2.2. Ces propriétés font partie des caractéristiques qui contribuent à la qualité de l'expérience de l'utilisateur.

L'analyse de ces résultats démontre ainsi l'intérêt du processus de découverte de services que nous proposons dans ce chapitre. Nous pensons que le mécanisme proposé permet réellement de sélectionner le service qui correspond au mieux aux besoins de l'utilisateur, et

ceci grâce à la fois à son approche intentionnelle, laquelle est plus transparente pour l'utilisateur, et à l'usage de contexte qui limite les services à ceux qui sont valides, c.-à-d. le contexte de validité et d'exécution du service. Toutefois, il est important de noter que nous ne pouvons obtenir cette bonne qualité des résultats que si le concepteur du système établit dès le départ une description riche des services disponibles et des différentes ontologies utilisées, d'où l'importance de la description des éléments de l'*espace de services* (*cf.* Chapitre 5).

7.4. CONCLUSION

Dans le cadre des SIP, l'accès aux services demeure trop complexe. En effet, dans un grand nombre de situations, l'utilisateur doit choisir tout seul un service parmi plusieurs implémentations disponibles, sans avoir pour autant le bagage nécessaire pour comprendre ces implémentations. Afin d'augmenter la transparence d'utilisation de ces systèmes, nous devons lui offrir le service qui satisfait ses besoins, sans qu'il soit forcé de comprendre les détails concernant l'implémentation ou encore les contraintes des dispositifs utilisés.

Dans ce chapitre, nous proposons une approche centrée sur l'utilisateur capable d'apporter à celui-ci des services adaptés à son intention et à son contexte d'utilisation tout en gardant un niveau de transparence convenable. Afin de démontrer cette approche, nous avons implémenté le processus proposé. Ce processus de découverte de services a ainsi été évalué selon deux critères : (i) le *passage à l'échelle*, qui nous permet d'analyser la faisabilité du processus proposé, notamment par rapport à la montée en charge des répertoires de services ; et la (ii) *qualité des résultats*, dans laquelle deux métriques majeures dans le domaine des services, la précision et le rappel, permettent d'analyser si le mécanisme proposé atteint réellement son objectif. Des mesures effectuées sur un ensemble très diversifié de machines nous permettent de démontrer la faisabilité de notre approche et nous incitent à la poursuite, notamment à travers la prédiction de services (*cf.* Chapitre 8).

Chapitre 8. PREDICTION DE SERVICES GUIDEE PAR L'INTENTION ET LE CONTEXTE

8.1. INTRODUCTION

De nos jours, les applications pour environnements pervasifs sont notamment des applications réactives basées uniquement sur le contexte courant de l'utilisateur. Un comportement *proactif et anticipatif*, notamment en prévision de la situation future de l'utilisateur, est peu développé. La plupart des recherches sur ce sujet reste à un niveau technique, anticipant une prochaine information contextuelle ou découvrant le service le plus approprié. Ces recherches ne considèrent pas les exigences intentionnelles relatives à l'expérience de l'utilisateur. En conséquence, plusieurs possibilités sont souvent fournies à l'utilisateur, même s'il n'est pas toujours en mesure de comprendre ce qui lui est proposé. Nous pensons que, afin d'atteindre la transparence prônée par Weiser (Weiser, 1991), les SIP doivent réduire l'effort de compréhension de l'utilisateur. Ils doivent cacher la complexité de ces multiples implémentations et de ces situations de contexte. Cela ne sera possible que grâce à une vision centrée utilisateur. Cette vision peut être atteinte grâce à un processus de prédiction permettant de mieux comprendre les besoins futurs de l'utilisateur afin de l'aider à y répondre d'une manière non intrusive. Ceci nous permettra d'offrir une meilleure pro-activité, permettant ainsi de contribuer à l'amélioration de la transparence nécessaire des SIP.

Nous proposons, dans cette nouvelle vision des SIP, une approche de prédiction intentionnelle et contextuelle. Notre but dans ce chapitre est de prédire l'intention future de l'utilisateur en fonction d'intentions précédentes dans des contextes similaires, afin de lui offrir le service le plus approprié qui peut l'intéresser d'une manière transparente et discrète. Nous supposons que l'utilisateur, dans son quotidien, réalise bien souvent et de manière répétitive une séquence d'intention dans un contexte donné. En observant ces séquences, nous pouvons être capable d'anticiper ses besoins à partir d'une intention initiale. L'objectif principal de cette approche est de fournir à l'utilisateur un service qui peut répondre à ses besoins futurs d'une manière non-intrusive, ce qui réduit l'effort de compréhension de l'utilisateur. En d'autres termes, nous proposons le développement d'un processus de prédiction de services, permettant de prédire, à partir de l'historique de l'utilisateur et à partir de son intention et de son contexte courant, les services répondant à son intention future, avant même qu'il ait besoin de l'exprimer.

Dans ce chapitre, nous présentons un processus de prédiction de services à la fois contextuel et intentionnel. Ce processus se base sur l'hypothèse que des situations similaires peuvent être détectées, surtout dans le cadre d'un Systèmes d'Information Pervasifs qui représente un environnement de travail dans lequel l'utilisateur peut avoir certaines habitudes. Ceci ne serait pas forcément le cas dans un contexte autre que d'un SIP. Nous présentons, par

la suite, une implémentation de ce mécanisme et analysons les résultats expérimentaux menés notamment par rapport au passage à l'échelle, à la précision et au rappel.

8.2. PROCESSUS DE PREDICTION DE SERVICES GUIDEE PAR LE CONTEXTE ET L'INTENTION

Nous observons que, dans le cadre d'un SIP, un utilisateur peut exprimer des intentions similaires formulées dans des contextes semblables. Il peut également suivre un ensemble d'habitudes qui évoluent avec le temps. Ceci reflète certaines habitudes de l'utilisateur lorsqu'il interagit avec son SIP à travers son *espace de services*. En exploitant ceci, nous pouvons améliorer la transparence des SIP en réduisant l'effort de l'utilisateur par l'anticipation de ces besoins. Ceci consiste à offrir à l'utilisateur, en toute transparence, le service qui répond à son intention future dans un contexte semblable. Ainsi, découle le besoin de présenter un nouveau processus de prédiction de services qui prend en considération les contextes et les intentions précédents de l'utilisateur.

Par conséquence, nous ici un processus de *prédiction de services*. Ce processus a comme objectif l'introduction des techniques de recommandation afin d'augmenter le caractère dynamique des SIP en suggérant à l'utilisateur, d'une manière transparente et discrète, le service le plus approprié qui pourra, par la suite, l'intéresser. L'objectif est de mieux comprendre les besoins de l'utilisateur afin de l'aider à y répondre d'une manière non intrusive. Il s'agit notamment d'observer le contexte dans lequel les utilisateurs évoluent et sollicitent certains services. A partir de ces observations, l'objectif est de pouvoir suggérer à ces utilisateurs les services les mieux adaptés qui conviennent à leurs intentions émergentes, dans un contexte similaire et avec des intentions semblables, voire de les leur proposer en anticipation. Ceci nous permettra d'offrir une meilleure pro-activité au système par une meilleure compréhension de la relation entre la notion de contexte et d'intention.

Nous proposons, ainsi, un processus de prédiction de l'intention future de l'utilisateur en se basant sur le contexte. Cette approche fournit de manière proactive un service qui peut répondre à ses besoins futurs. En effet, cette approche est basée sur l'hypothèse que des situations ordinaires $\mathcal{S}$ peuvent être détectées. Sur la base de cette hypothèse, ce mécanisme de prédiction considère un ensemble de séries temporelles représentant la situation observée de l'utilisateur. Nous définissons la notion de situation $\mathcal{S}$ comme étant l'intention de l'utilisateur I_U, dans un contexte donné Cx_U, satisfaite par un service spécifique Sv_i. Ces situations sont horodatées et stockées dans une base de données après chaque processus de découverte de services. Ceci représente l'historique $\mathcal{H}$. Ainsi, en analysant l'historique, le mécanisme de prédiction peut établir un modèle de comportement de l'utilisateur $\mathcal{M}c$ dans un environnement dynamique et en déduire immédiatement son intention future.

Notre proposition de prédiction de services guidée par l'intention et le contexte se compose, comme l'illustre la Figure 66, de deux principaux processus à savoir :

- le ***processus d'apprentissage*** dans lequel des situations similaires $\mathcal{S}$ sont regroupées en *clusters*, lors de l'étape de *clustering*. Un *cluster* représente un ensemble de situations, rassemblées dans un même groupe, présentant des similarités entre elles en termes d'intention et de contexte. C'est une façon de détecter et de regrouper des situations récurrentes et similaires. Dans l'étape suivante, ces *clusters* sont interprétés comme des *états* d'une machine à états. Les probabilités de transition d'un état à un autre sont alors calculées en fonction de l'historique. Cette étape, appelée étape de *classification*, vise à représenter, à partir des *clusters* reconnus, le modèle de comportement de l'utilisateur $\mathcal{M}c$ en fonction de ses situations $\mathcal{S}$. Un *modèle de comportement* schématise les habitudes des utilisateurs en fonctions de ses situations dans le passé. Il illustre ses habitudes en fonctions d'un ensemble d'états (qui représentent les *clusters* reconnus lors de la phase précédente) avec une probabilité de transition d'un état vers un autre, c.-à-d. la probabilité d'avoir l'intention future $I2$ à partir de l'intention immédiate $I1$ dans un contexte semblable. En interprétant le changement de situations en tant que trajectoire d'états, nous pouvons anticiper les besoins futurs de l'utilisateur. Dans notre approche, ce processus consiste à estimer les probabilités de passer d'une situation à d'autres situations futures possibles.
- Le ***processus de prédiction*** se base sur le modèle de comportement de l'utilisateur $\mathcal{M}c$, sur l'intention immédiate de l'utilisateur $I_{\mathcal{U}}$ et sur son contexte courant $Cx_{\mathcal{U}}$. Sur la base des probabilités de $\mathcal{M}c$, calculées dans le processus précédent, ce processus se charge d'estimer la prochaine intention, et donc le prochain service. Ainsi, il fournit à l'utilisateur un service qui peut répondre à ses besoins futurs d'une manière transparente.

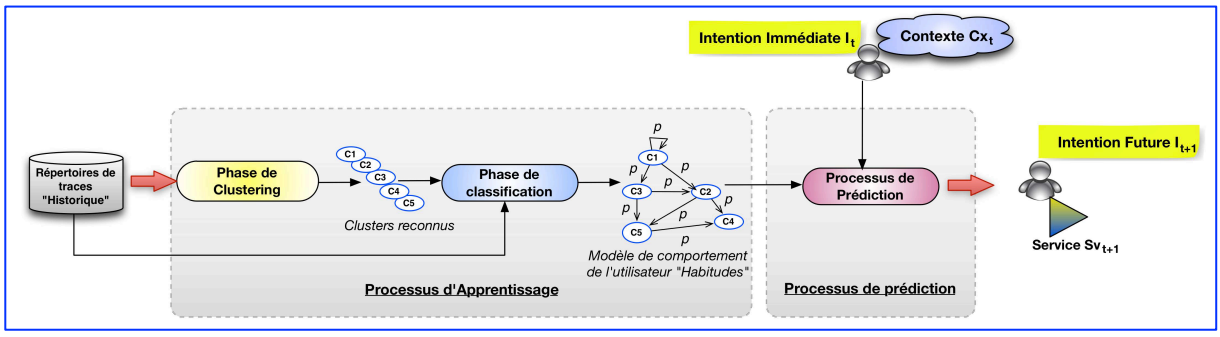


Figure 66. La prédiction de services selon l'intention et le contexte de l'utilisateur

Avant de détailler ces processus d'apprentissage et de prédiction, nous devons d'abords considérer la structure de l'historique utilisé par ces processus. Il s'agit de la gestion des traces, décrite dans la section suivante.

8.2.1. La gestion des traces (historiques)

Dans le cadre de ce travail, l'*historique* $\mathcal{H}$ est alimenté par tous les résultats obtenus par le processus de découverte de services, formant ainsi les traces utilisées pour la prédiction. Le processus de découverte de services est basé sur l'intention et le contexte de l'utilisateur afin de trouver les services les plus appropriés. Le service, qui répond au mieux à l'intention immédiate de l'utilisateur dans son contexte courant, est sélectionné. Nous définissons la notion de situation S_i comme la représente la Définition 10 :

Définition 10 :

Une situation S_i est définie par une intention I_υ, qui caractérise l'intention de l'utilisateur lors de sa demande d'un service, par un contexte Cx_υ, qui représente son contexte courant à cet instant donné, et par le service sv_i sélectionné pour cette démarche lors du processus de découverte de services.

$$S_i = < I_\upsilon,\ Cx_\upsilon,\ sv_i >$$

Définition 10. Formalisation de la notion de situation

Le mécanisme de prédiction de l'intention future est fondé non seulement sur la situation actuelle de l'utilisateur, mais aussi sur ses situations observées précédemment. Afin de pouvoir construire le modèle de comportement de l'utilisateur (*cf.* section 8.2.2.2), ces situations doivent être sauvegardées. Ces situations observées, sont enregistrées comme des séries temporelles formant ainsi l'historique de l'utilisateur $\mathcal{H}$. Chaque série temporelle représente une situation S_i horodatée, comme l'illustrent le Tableau 5.

Tableau 5. Structure de l'historique de l'utilisateur $\mathcal{H}$

Temps/Date	Intention	Contexte	Service
t_1	I_{υ_1}	Cx_{υ_1}	Sv_1
t_2	I_{υ_2}	Cx_{υ_2}	Sv_2
t_3	I_{υ_3}	Cx_{υ_3}	Sv_3
…	…	…	…
t_i	I_{υ_i}	Cx_{υ_i}	Sv_i
…	…	…	…
t_n	I_{υ_n}	Cx_{υ_n}	Sv_n

Quand un service est sélectionné, après un processus de découverte de services, la *situation de l'utilisateur* S_i est enregistrée à la fin dans l'historique $\mathcal{H}$ afin de garder une trace des situations passées de l'utilisateur. L'*intention* I_υ est représentée suivant un schéma XML contenant deux éléments obligatoires, à savoir le *verbe* et la *cible*. Comme nous l'avons

mentionné dans la section 7.2.2.3, nous avons choisi de représenter l'intention uniquement sous forme de ces deux éléments puisqu'ils représentent les éléments de base d'une intention et qu'ils sont obligatoires lors de sa représentation. Toutefois, nous avons choisi de ne pas considérer les paramètres dans nos processus de découverte et de prédiction de services vu que le nombre de paramètres de nature différente à prendre en compte rend la description de l'intention bien trop précise et pas assez générale. De plus, en prenant en compte ces paramètres lors de ce processus, les services retournés comme les plus appropriés vont être trop spécifiques. Il est à noter également que la description des intentions suit le même schéma que celui présenté dans la section 6.3.1.3. Ensuite, le *contexte* $Cx_{\mathcal{U}}$ est également représenté selon le schéma XML présenté dans la section 6.4.2, contenant la description d'un contexte précis celui observé lors que l'intention a été demandée. Enfin, le *service* $\mathcal{S}v_i$ représente l'identifiant du service sélectionné pour satisfaire cette intention dans ce contexte.

Dans l'historique, les traces représentent des situations de l'utilisateur ($\mathcal{S}_i$) enregistrées à un instant donné. En s'inspirant des observations de contexte (O_{cpi}) (*cf.* Définition 4), nous introduisons la notion d'observation de situations (O_{Si}), comme définie dans la Définition 11, représentant une situation de l'utilisateur $\mathcal{S}_i$ observée à l'instant t_i.

Définition 11 :

Une observation de situation O_{Si} est définie comme étant la situation de l'utilisateur $\mathcal{S}_i$ qui a été observée et stockée dans l'historique $\mathcal{H}$ à un instant t_i

$$O_{Si} = \{<\mathcal{S}_i, t_i> \mid \forall\ i \in [1, n], \mathcal{S}_i \in \mathcal{H} \wedge TimeStamp\ (\mathcal{S}_i) = t_i\}$$

Définition 11. Formalisation de la notion d'observation de situation

Nous définissons ainsi l'historique de l'utilisateur $\mathcal{H}$ comme l'illustre la Définition 12.

Définition 12 :

Un historique $\mathcal{H}$ est représenté par l'ensemble de toutes les observations de situations O_{Si} classées en fonction de leurs occurrences.

$$\mathcal{H} = \{O_{Si}\}$$

$i \in [1, n]$, avec n représente la taille de l'historique $\mathcal{H}$

Définition 12. Formalisation de la notion d'historique

Le maintien des traces des situations observées de l'utilisateur (O_{Si}) est nécessaire au processus d'apprentissage afin de déduire le comportement de l'utilisateur. Ce processus d'apprentissage sera expliqué dans la section suivante.

8.2.2. Le processus d'apprentissage

Afin de permettre un comportement anticipatif et proactif de la part des SIP, nous devons d'abord apprendre de manière dynamique le comportement de l'utilisateur. Ceci est une étape importante pour le mécanisme de prédiction.

Le processus d'apprentissage est basé sur l'analyse de l'historique $\mathcal{H}$. Nous procédons en regroupant les différentes observations O_{Si} dans des *clusters* ($\mathcal{CL}$) qui représentent des situations similaires (phase de *clustering, cf.* section 8.2.2.1). L'historique peut contenir des situations similaires mais exprimées avec des contextes incertains ou avec des intentions semblables exprimées de différentes manières (variabilité dans l'expression de l'intention, *cf.* section 6.3). Le rôle du *clustering* ici est de prendre en considération cette ambiguïté en représentant ces situations, qui sont similaires, dans un même groupe. Ainsi, les *clusters* sont plus pertinents à traiter que des situations séparées.

Par la suite, le processus d'apprentissage se charge d'apprendre le *modèle de comportement de l'utilisateur*, illustrant ses habitudes, à partir des *clusters* reconnus lors de la dernière étape et de l'historique. Les *clusters* vont former les *états* de ce modèle de comportement. Les probabilités de transition d'un état à un autre sont ainsi calculées à partir de l'historique. Ceci représente la phase de *classification* (cf. section 8.2.2.2).

Le processus d'apprentissage est déclenché indépendamment de l'étape de prédiction, et peut être caractérisé comme une tâche d'arrière-plan qui s'exécute périodiquement. Nous présentons les deux étapes de *clustering* et de *classification* dans les deux sections suivantes.

8.2.2.1. Phase de clustering

La première étape de notre mécanisme de prédiction intentionnel et contextuel est le regroupement des traces de l'utilisateur. Ces traces, comme nous les avons définies dans la section 8.2.1, représentent les observations O_{Si} des situations S_i stockées dans l'historique $\mathcal{H}$. Puisque l'utilisateur interagit quotidiennement avec son SIP, qui représente son environnement de travail, certaines de ces situations peuvent être récurrentes. Ces situations récurrentes peuvent être exprimées avec des contextes incertains ou avec une certaine variabilité d'expression des intentions (*cf.* section 6.3). Le rôle du *clustering* ici est de prendre en considération la pertinence de ces situations, qui peuvent être classées/groupées par similarités dans des *clusters*. En outre, l'analyse des *clusters* permet une meilleure représentation des habitudes de l'utilisateur, puisqu'ils sont plus pertinents à traiter que des situations séparées. Ceci peut améliorer la précision de notre mécanisme de prédiction.

L'entrée de cette étape de *clustering* est un ensemble de vecteurs représentants les observations de l'utilisateur stockées dans l'historique (*cf.* Tableau 5). Le rôle principal du *clustering*, comme l'illustre la Figure 67, est de détecter les observations récurrentes parmi toutes les situations observées auparavant et de les regrouper en *cluster*. Un *cluster* est constitué d'un *centroïde* et d'un ensemble d'observations. Le *centroïde* représente l'identifiant du *cluster* lequel symbolise l'observation la plus similaire à toutes les observations groupées dans ce *cluster*. Ce *centroïde* est définit par le triplet <intention ($\mathcal{I}$), contexte ($\mathcal{C}x$), service ($\mathcal{S}v_i$)>. Ainsi, pour chaque observation du vecteur d'entrée, la phase de *clustering* se charge de déterminer le *centroïde du cluster* qui est le plus similaire à cette observation. Cette similarité correspond, en réalité, à des intentions très similaires dans des contextes semblables. Un utilisateur peut exprimer la même intention d'une manière légèrement différente en utilisant des verbes et des cibles qui sont sémantiquement assez similaires. Basé sur les ontologies des verbes et des cibles, nous effectuons une *mise en correspondance intentionnelle* afin de déterminer leur degré de similarité (*cf.* section 7.2.2.3). Le contexte de l'utilisateur représente quant à lui des données très hétérogènes : numérique, symbolique, qualitative, etc. La même classe d'éléments de contexte peut avoir des représentations différentes (par exemple, l'emplacement peut être exprimé sous forme de coordonnées GPS, d'adresse postale, d'emplacement prédéfini, etc.). Ainsi, pour comparer deux descriptions de contexte, nous utilisons également *une mise en correspondance contextuelle* qui détermine la similarité entre les observations des descriptions contextuelles. Cette mise en correspondance contextuelle diffère de celle présentée dans la section 7.2.2.4. Lors de la découverte de services, nous évaluons la satisfaction des différentes conditions du contexte requis du service ($\mathcal{C}x\mathcal{R}$) par les observations du contexte courant de l'utilisateur ($\mathcal{C}x\mathcal{U}$). Cependant, dans le cadre du processus de prédiction nous comparons la similarité entre les observations contextuelles des utilisateurs prises à des instants donnés.

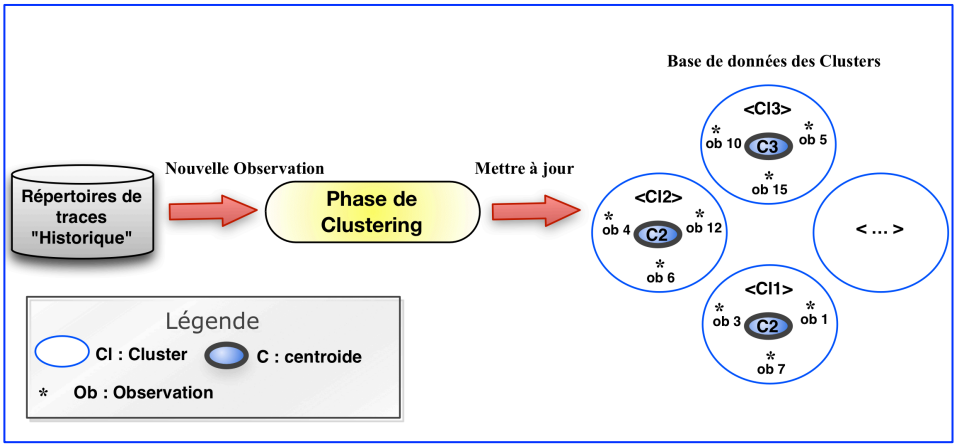


Figure 67. Mise à jour et création des *clusters* durant la phase de *clustering*

La phase de *clustering* permettra de trouver ces situations et de les représenter par une situation commune qui est la plus proche de tous les membres d'un même *cluster*. Toutefois, afin de mieux s'adapter aux SIP, l'algorithme de *clustering* doit répondre à certaines

exigences. Nous nous basons sur les exigences de Mayrhofer (Mayrhofer, 2004) (*cf.* section 3.5.3.1.1). Parmi des critères, certains nous semblent essentiels pour les SIP :

- ***Non-supervisé*** : les *clusters* doivent être formés de manière automatique, sans connaissance préalable et sans l'aide de l'utilisateur ;
- ***Mise à jour* :** le processus de *clustering* doit mettre à jour les *clusters* déjà reconnus étant donné que le comportement de l'utilisateur peut changer ;
- ***Hors ligne*** : les *clusters* doivent être mis à jour régulièrement, sans entrave au fonctionnement normal du système, ce qui suggère une stratégie *« hors ligne »*. Cela peut être basé sur un paramètre de *clustering* qui définit à quel moment ce processus sera enclenché. Ce paramètre peut être défini en fonction de la dynamique des systèmes qui les emploient ;
- ***Confidentialité*** : le processus de *clustering* doit prendre en compte la protection de certaines informations personnelles ;

Dans le cadre des SIP, les situations des utilisateurs peuvent évoluer avec le temps. Il est difficile ainsi de prévoir au préalable les *clusters* qui doivent être formés. En conséquence, notre algorithme de *clustering* doit être *non-supervisé* déterminant de manière automatique, sans connaissance préalable et sans l'intervention de l'utilisateur, les *clusters* regroupant des situations de l'utilisateur. De plus, il doit *mettre à jour* ces *clusters* déjà reconnus afin de prendre en considération de nouvelles situations ainsi que l'évolution et le changement dans les situations de l'utilisateur. Par ailleurs, l'algorithme de *clustering* doit être mis *hors ligne*. Il se déclenchera à partir d'un certain moment qui doit être défini au préalable et qui sera pris en compte lors de l'exécution. Ceci est dans le but de collecter un certain nombre de traces à traiter, d'une part, et de réduire les coûts durant la mise à jour des *clusters*, d'autre part. Par la suite, nous devons tenir compte du fait que l'utilisateur pourrait souhaiter que certaines informations de contexte ne soient pas utilisées dans le processus de *clustering*. C'est pour cette raison que le critère de *confidentialité* devrait être pris en compte dans ce processus.

Tableau 6. Comparaison des algorithmes de *clustering*

	KM (*K-Means)*	**FKM** *(Fuzzy K-Means)*	**SOM** (*Self-Organizing Map*)	**NG** (*Neural Gas*)	**GNG** (*Growing Neural Gas*)
Hors ligne					
Mise à jour	✓		variant		✓
Confidentialité	✓	✓	✓	✓	✓
Exécution en temps réel	---	---	--	-	-
Clusters illimités					✓

Différent algorithmes de *clustering* existent dans la littérature. Le Tableau 6 présente une comparaison entre certains de ces différents algorithmes, en se basant sur l'évaluation de

Mayrhofer (Mayrhofer, 2004). A partir du Tableau 6, nous pouvons observer que les algorithmes *K-Means* (KM) (Daszykowski et al., 2002) et *Fuzzy K-Means* (FKM) (Nelles, 2001) ne peuvent pas être appliqués dans notre cas. En effet, ils exigent une connaissance préalable sur les *clusters* durant la phase d'apprentissage et demandent un temps d'exécution relativement élevé. En outre, l'algorithme FKM échoue sur les aspects de mise à jour puisqu'il n'adapte pas dynamiquement les *clusters* déjà reconnus aux changements. L'algorithme NG *(Neural Gas)* (Martinetz et al., 1993), quant à lui, nécessite une spécification préalable du nombre de *clusters* à utiliser. Cette contrainte a conduit à l'élimination de l'algorithme NG, car il est difficile de déterminer le nombre de *clusters* a priori dans un environnement dynamique pervasif. En outre, SOM (*Self-Organizing Map*) (Kohonen, 1995) peut également être supprimée pour les mêmes raisons que le *K-Means*. Par conséquent, l'algorithme GNG (*Growing Neural Gas*) (Daszykowski et al., 2002) semble être le candidat le plus approprié, car il correspond le plus aux critères cités ci-dessus. Il s'adapte aux changements, ne nécessite pas de connaissances préalables et a une exécution en temps réel raisonnable. Le rôle de GNG (Daszykowski et al., 2002) consiste à reconnaître et mettre à jour un ensemble de *clusters* en fonction du vecteur d'entrée. Il relie l'entrée à un ensemble de nœuds de sortie que nous avons appelé *« clusters »*. Toutefois, l'algorithme GNG représente un algorithme très complexe à réutiliser et demande un opérateur de normalisation afin de coder (en numérique) toutes les entrées qui sont de types hétérogènes. Ainsi, la réutilisation de cet algorithme demande beaucoup d'effort de mise en œuvre. Pour cette raison, nous avons choisi de concevoir et développer notre propre algorithme de clustering, en s'inspirant de ces algorithmes et en se basant la mise en correspondance implémentée dans l'algorithme de découverte de services.

8.2.2.1.1. Principe de l'algorithme de clustering

L'objectif principal de l'algorithme de *clustering*, comme l'illustre la Figure 68, est de permettre de déterminer si une observation O_S appartient à un cluster déjà existant ($C\mathcal{L}_i$) ou si elle déclenchera la création d'un nouveau *cluster* le contenant ($C\mathcal{L}_{ii}$). Pour ce faire, cet algorithme compare sémantiquement l'observation O_{Si} avec le centroïde de chaque *cluster* déjà reconnu ($C\mathcal{L}_i$). Cette comparaison se base sur une mise en correspondance intentionnelle entre l'intention de l'observation et celle du centroïde, et d'une mise en correspondance contextuelle entre les descriptions contextuelles de ces deux derniers, qui vont définir à la fin le degré de similarité. En suite, il déclenche une fonction *getBestCluster*. Cette fonction se charge de déterminer le *cluster* dont le centroïde a obtenu le score de mise en correspondance le plus élevé avec l'observation O_{Si}. De plus, cette fonction vérifie que ce score obtenu excède un certain seuil λ qui doit être introduit au préalable par le concepteur du système. Ce seuil est le seuil d'acceptation d'une nouvelle observation dans un *cluster*. Cependant, si le score est inférieur à ce seuil alors il n'est pas possible de rajouter l'observation au *cluster* retenu (même si son score est le plus élevé), mais elle sera introduite dans un nouveau *cluster*.

Pour cela, cet algorithme commence par récupérer le contexte (Cxo_{Si}) (ligne 5), l'intention (Io_{Si}) (ligne 6) et le service (Svo_{Si}) (ligne 7) de l'observation O_{Si}. Ensuite, pour tous les *clusters* déjà reconnus ($C\mathcal{L}$), l'algorithme va calculer le score de mise en correspondance entre

l'observation O_{Si} et chaque centroïde du *cluster* CL_i de CL (ligne 8-15). Il commence par calculer le score de la mise en correspondance intentionnelle (I_{score}) entre Io_{Si} et Icl_i (ligne 11), ainsi que le score de la mise en correspondance contextuelle (C_{score}) entre Cxo_{Si} et $Cxcl_i$ (ligne 12). Le score final de la mise en correspondance (CL_{score}) sera calculé en fonction de ces deux derniers scores, selon la formule suivante : $CL_{score} = (I_{score} + C_{score}) / 2$ (ligne 13). Le couple $<CL_i, CL_{score}>$ est, par la suite, rajouté à la liste CL_{ranked} des *clusters* traités (ligne 14).

Procedure *clustering* (O_{Si}, CL, $OntoT$, $OntoV$, $OntoCX$)

(1) $I_{score} = 0$
(2) $C_{score} = 0$
(3) $CL_{score} = 0$
(4) $CL_s = \emptyset$ /* représente les clusters à la sortie de cet algorithme */
(5) Cxo_{Si} = *GetContext* (O_{Si})
(6) Io_{Si} = *GetIntention* (O_{Si})
(7) Svo_{Si} = *GetService* (O_{Si})

(8) **For each** $CL_i \in CL$ **Do**
(9) $Cxcl_i$ = *GetContext* (CL_i)
(10) Icl_i = *GetIntention* (CL_i)
(11) I_{score} = *IntentionMatching* (Io_{Si}, Icl_i, $OntoT$, $OntoV$) /* Mise en correspondance intentionnelle */
(12) C_{score} = *ContextMatching* (Cxo_{Si}, $Cxcl_i$, $OntoCX$) /* Mise en correspondance contextuelle */
(13) $CL_{score} = (I_{score} + C_{score}) / 2$
(14) $CL_{ranked} = CL_{ranked} \cup \{< CL_i, CL_{score} >\}$
(15) **End For**
(16) CL_{best} = *getBestCluster* (CL_{ranked}) /* détermine le *cluster* qui a obtenu le degré de similarité le plus élevé et qui dépasse un seuil α*/
(17) **If** CL_{best} = null **Then**
(18) CL_{ii} = *CreateNewCluster* (O_{Si}) /* avec un Centroïde < Io_{Si}, Cxo_{Si}, Svo_{Si} > */
(19) CL = *addNewCluster* (CL, CL_{ii})
(20) **Else**
(21) CL = *updateCluster* (CL, CL_{best}) /* rajouter l'observation O_S à la liste des observations de CL_i */
(22) **End If**

(23) **Return** CL
(24) **End Procedure**

Figure 68. Algorithme de *clustering* des observations de l'historique $\mathcal{H}$

Ensuite, dès que tous les *clusters* seront traités, l'algorithme lance la vérification si l'observation O_{Si} est similaire à un *centroïde* des *clusters* dans la liste CL_{ranked}, ou s'il fera l'objet d'un nouveau *cluster* CL_{ii} (ligne 17-22). Cette étape procède, grâce à la fonction *getBestCluster* (ligne 16), par la récupération du *cluster* CL_{best} ayant eu le score CL_{score} le plus

élevé de la liste CL_{ranked} (ligne 16) et qui dépasse le seuil λ. Si aucun *cluster* n'est déterminé par cette fonction, alors l'algorithme conclut que O_{Si} ne peut pas se rajouter à aucun *cluster* de CL. Dans ce cas, O_{Si} sera attribué à un nouveau *cluster* CL_{ii} (ligne 18), grâce à la fonction *createNewCluster*. Cette fonction va créer un nouveau *cluster* CL_{ii} qui aura comme centroïde le triplet $<Io_{Si}, Cxo_{Si}, Svo_{Si}>$ et comme observation attachée à ce *cluster* l'observation O_{Si}. Ce nouveau *cluster* CL_{ii} est, par la suite, rajouté à la liste des *clusters* CL (ligne 19). Sinon, si un *cluster* CL_i est sélectionné, alors l'algorithme va mettre à jour la liste des observations attachées au *cluster* CL_i en ajoutant l'observation O_{Si} (ligne 21).

Les *clusters* reconnus, lors de cette phase, évoluent avec le temps, certains *clusters* disparaissent et d'autres se mettent à jour. Ainsi, il est essentiel de prendre en compte ces changements dans une phase de maintenance des *clusters*. Nous envisageons de prendre en considération tout changement dans l'historique. Par exemple, si une observation est supprimée de l'historique (considérée comme trop ancienne, inutile, etc.), nous supprimons cette observation du *cluster* qui la regroupe. De plus, ces *clusters* sont utilisés par la suite dans la phase de classification. Ainsi, nous estimons qu'un *cluster* qui n'est pas pris en compte pour représenter les habitudes de l'utilisateur, est un *cluster* peu pertinent. Cette pertinence peut être déterminée à partir d'un paramètre de *« refreshness »* qui va être décrémenté à chaque fois que l'algorithme de classification ne met pas à jour ce *cluster*, et incrémenté dans le cas contraire. A partir d'un certain seuil, déterminé par le concepteur lors de la phase de conception, ce *cluster* sera considéré comme peu pertinent et sera éliminé par la suite. Cette phase de maintenance représente une de nos perspectives à court terme.

Ensuite, une fois le processus de *clustering* terminé, les *clusters* reconnus sont ensuite interprétés comme des états du modèle de comportement de l'utilisateur mis à jour par le processus de classification, présenté dans la section suivante.

8.2.2.1.2. Complexité de l'algorithme de clustering

Afin de déterminer la complexité de notre algorithme de *clustering* guidé par l'intention et le contexte, nous avons reformulé notre algorithme comme suit :

Procédure Clustering

Initialisation des variables Iscore, Cscore, CLscore, CLs

Début
Pour chaque *ensemble S1 (boucle 1)*
Calculer la mise en correspondance intentionnelle (Iscore)
Pour chaque *ensemble S2 (boucle 2)*
Pour chaque *ensemble S3 (boucle 3)*
Calculer la mise en correspondance contextuelle (Cscore)
Fin pour

Fin pour
Calculer le score final de mise en correspondance (Sscore)
Fin pour
Créer ou mettre à jour le cluster
Retourner le nouveau cluster ou mis à jour
Fin

A l'instar de l'algorithme de découverte de services (*cf.* section 7.2.2.5), cet algorithme se base sur les trois ensembles : (i) S1 ⊂ {1, 2, 3, ⋯, N} représente l'ensemble des *clusters* stockées dans base de données ; (ii) S2 ⊂ {1, 2, 3, ⋯, M} représente l'ensemble des observations contextuelles décrites dans le contexte de l'observation ; et (iii) S3 ⊂ {1, 2, 3, ⋯, L} représente l'ensemble des observations contextuelles décrites dans le contexte du centroïde du *cluster*. Cet algorithme commence par traiter la boucle externe 1. Cette boucle s'exécute N fois et fais appel à chaque fois à la boucle interne 2 qui s'exécute à son tour M fois. Ensuite, à chaque fois que la boucle 2 s'exécute, la boucle interne 3 s'exécute L fois. En conséquence, le traitement de la boucle 3 se lance au total N*M*L fois. Ainsi, la complexité de cet algorithme est O(N*M*L). Nous admettons que N est le nombre maximum. Ainsi, dans un cas particulier où la condition d'arrêt de la boucle interne 2 est inférieure à N au lieu d'être inférieure à M et la condition d'arrêt de la boucle interne 3 est également inférieure à N au lieu d'être inférieure à L, la complexité totale pour les trois boucles est $O(N^3)$. Ainsi, nous pouvons déterminer que l'algorithme de *clustering* guidé par l'intention et le contexte est un algorithme de complexité polynomiale de degré trois.

8.2.2.2. Phase de classification

Dans le cadre d'un SIP, les utilisateurs suivent un modèle de comportement qui représente un modèle mathématique reflétant leurs dynamiques, *i.e.* leurs habitudes. L'utilisateur ne peut pas être décrit avec précision de façon préalable. Par conséquent, un modèle dynamique de comportement de l'utilisateur est nécessaire. Il doit être capable de s'adapter aux changements de l'utilisateur et de prendre en considération la nature probabiliste de son comportement.

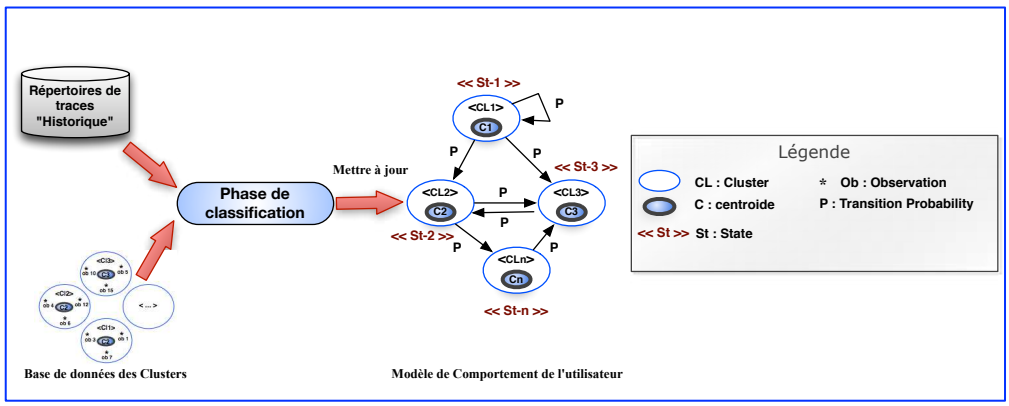


Figure 69. Mise à jour et Création du modèle de comportement de l'utilisateur durant la phase de classification

A partir des *clusters* reconnus lors de la phase précédente de *clustering* et de l'historique de l'utilisateur, le module de *classification* détermine et maintient le modèle de comportement de l'utilisateur, comme l'illustre la Figure 69. Ce modèle représente le comportement de l'utilisateur comme un ensemble d'*états* avec une probabilité de transition. Chaque *état* est représenté par le centroïde d'un *cluster*. Chaque probabilité est calculée en fonction de l'historique et détermine la probabilité de passer d'un état à un autre.

Comme pour les algorithmes de *clustering*, les algorithmes de classification présentent certaines exigences lorsqu'ils sont appliqués aux SIP. Ces algorithmes doivent s'adapter à l'évolution et à la dynamique des environnements pervasifs. Dans un tel environnement, il est difficile d'établir une connaissance préalable relative au comportement de l'utilisateur. Ainsi, inspiré des travaux de Mayrhofer (Mayrhofer, 2004) et de Sigg (Sigg, 2008), nous pouvons citer, parmi les conditions nécessaires pour une meilleure classification dans un environnement pervasif, les conditions suivantes :

- **non supervisé :** le modèle doit être estimé de manière automatique, sans une connaissance préalable et sans l'aide de l'utilisateur ;
- **Incrémental :** quand un nouveau *cluster* est reconnu, le modèle de comportement de l'utilisateur doit augmenter progressivement sa structure interne, sans nécessiter un apprentissage à temps plein ;
- **Données hétérogènes et multidimensionnelles :** les situations de l'utilisateur sont représentées par des données hétérogènes qui peuvent être nominales, ordinaires, numériques, etc. Ces différents types des données doivent être prises en compte ;
- **Mémoire et processus de chargement :** dans un SIP, un algorithme de classification peut être déployé sur différents appareils mobiles avec, souvent, une capacité de mémoire limitée.

Plusieurs techniques de classification existent. Parmi ces techniques, nous notons le réseau Bayésien *(BN – Bayesian Network)* (Friedman et al., 1997), les chaînes de Markov (Feller, 1968), le modèle de Markov caché *(HMM – Hidden Markov Chain)* (Rabiner, 1989), ARMA (Hsu et al., 1998), les machines à vecteurs de support *(SVM - Support Vector Machines)* (Burges, 1998), ainsi que *Active Lempel Ziv* (ALZ) (Gopalratnam et Cook, 2003). Tout d'abord, l'algorithme BN (Friedman et al., 1997) fonctionne avec des variables discrètes. Il nécessite une connaissance préalable et doit préciser, dès le début, les différents états et variables cachées. Ceci ne remplit pas les conditions de classification nécessaires dans un SIP, citées ci-dessus. C'est aussi le cas de SVM (Burges, 1998). Ce dernier représente une méthode de classification ne traitant que des données numériques. Cette méthode demande, en plus, une taille fixe de l'espace des données d'entrée. L'algorithme ARMA (Hsu et al., 1998), quant à lui, représente l'une des techniques de classification les plus efficaces et les plus appropriées dans notre domaine. Néanmoins, l'inconvénient majeur est sa limitation au traitement des données numériques, ce qui rend difficile son application à l'intention et à d'autres données contextuelles ayant un caractère symbolique. Le HMM (Rabiner, 1989) représente une technique de classification bien reconnue. Cependant, celle-ci ne peut s'appliquer dans un environnement pervasif en raison de sa méthode supervisée.

Ainsi, l'algorithme de chaînes de Markov (Feller, 1968) nous semble plus adapté notamment par sa caractéristique *non supervisée*. En outre, les chaînes de Markov sont capables de classer les données multidimensionnelles et hétérogènes, ce qui rend cette méthode la plus appropriée pour les SIP en fonction des critères énoncés ci-dessus. Les chaînes de Markov (Feller, 1968) sont utilisées pour représenter un processus stochastique en temps discret avec un espace d'état discret $\{X_t : t = 0, 1,2 ,\ldots\}$. On représente le modèle des chaînes de Markov $\mathcal{M}c$ comme la paire $\mathcal{M}c = (\mathcal{S}t, p)$, avec $\mathcal{S}t$ représentant les différents états et $p \in [0,1]$ représentant la probabilité de transition d'un état à un autre (Feller, 1968). Dans ce modèle, l'état actuel ne dépends que de l'état précédent.

Dans notre cas, à un instant t_i donné, l'utilisateur est dans un *état* $\mathcal{S}t_i$ représentant son intention dans un contexte donné. Dans un SIP, l'intention de l'utilisateur ainsi que le contexte peuvent changer. Par conséquent, l'utilisateur se déplace d'un état $\mathcal{S}t_i$ vers un autre état $\mathcal{S}t_j$. L'état $\mathcal{S}t_j$ représente l'état qui succède l'état $\mathcal{S}t_i$ avec une probabilité p. Cette probabilité de transition représente le nombre de transition de $\mathcal{S}t_i$ à $\mathcal{S}t_j$ (N_{St_i,St_j}) divisé par le nombre de transitions possibles à partir de $\mathcal{S}t_i$ (N_{St_i,St_k}). Cette probabilité est représentée, par la Formule 8, comme suit :

$$p_{St_iSt_j} = P(X_{t+1} = St_j | X_t = St_i) = \frac{N_{St_iSt_j}}{N_{St_iSt_k}}$$

- p correspond à la probabilité de transition de St_i vers St_j
- St_i correspond à l'état de départ dans le modèle $\mathcal{M}c$
- St_j correspond à l'état d'arrivée à partir de St_i dans le modèle $\mathcal{M}c$
- St_k correspond aux différents états d'arrivée possibles à partir de St_i dans le modèle $\mathcal{M}c$
- X_t et X_{t+1} corresponds aux états de la chaîne de markov $\mathcal{M}c$
- N_{St_i,St_j} correspond au nombre de transition de St_i à St_i
- N_{St_i,St_k} correspond au nombre de transition de St_i à St_k

Formule 8. Probabilité de transition dans un modèle de chaines de Markov

La Figure 70 présente notre algorithme de classification (chaîne de Markov) lancé sur un ensemble d'observations de l'historique (O_S) afin de mettre à jour le modèle de comportement ($\mathcal{M}c$). Ce modèle de comportement représente dynamiquement les habitudes des utilisateurs sous forme d'états (ses situations représentées par les centroïdes des *clusters*) et de probabilités de transition (p) d'un état à un autre. Cette probabilité de transition va permettre de déterminer à partir de l'intention immédiate de l'utilisateur (représentée dans le triplet représentant le centroïde du *cluster*, *cf.* section 8.2.2.1), son intention future et par la suite le service le plus approprié qui peut l'intéresser. Il est à noter que cet algorithme de chaînes de Markov est un algorithme à complexité polynomiale de degré deux $O(N^2)$ (Hou et al., 2002).

Procedure classification (O_S, $\mathcal{M}c$)

(1) $MapPossibleSituation = \emptyset$ /* *Contient tous les couples d'observations considérées comme successives dans l'historique* */
(2) $MapNewTransition = \emptyset$ /* *Contient les couple d'observations considérées comme successives avec leur probabilité de transition de la première vers la deuxième* */
(3) $CLO_{Si} = \emptyset$
(4) $CLO_{Sii} = \emptyset$
(5) O_{Sii} = null /* *représente l'observation qui succède O_{Si} dans l'historique* */

(5) **For each** $O_{Si} \in O_S$ **Do**
(6) O_{Sii} = *GetNextObservation* (O_{Si}) /* *récupère l'observation O_{Sii} qui succède directement O_{Si} dans l'historique*/
(7) **If** TimeStamp (O_{Sii}) – TimeStamp(O_{Si}) < = μ **Then** /* *vérifier si les observations sont enregistrées dans l'historique à un intervalle de temps qui ne dépasse pas une durée déterminée par le seuil μ* */
(8) CLO_{Si} = *getCluster* (O_{Si}) /* *récupère l'identifiant du cluster auquel appartient O_{Si}* */
(9) CLO_{Sii} = *getCluster* (O_{Sii}) /* *récupère l'identifiant du cluster auquel appartient O_{Sii}* */
(10) $MapPossibleSituation = MapPossibleSituation \cup \{< CLO_{Si}, CLO_{Sii}>\}$
(11) **End If**
(12) **End For**

/* *Mettre à jour le modèle de comportement $\mathcal{M}c$ avec les nouvelles transitions possibilités et les nouvelles probabilités de transition d'un état à un autre*/

(13) $MapNewTransition = getNewTransition$ ($MapPossibleSituation$)
(14) $\mathcal{M}c = UpdateMarkovChainModel$ ($MapNewTransition$, $\mathcal{M}c$)

(15) **Return** $\mathcal{M}c$
(16) **End Procedure**

Figure 70. Algorithme de classification basé sur les chaines de Markov

L'algorithme de classification prend en entrée un ensemble d'observations de l'historique (O_S) et le modèle de comportant de l'utilisateur ($\mathcal{M}c$). Pour la première application de cet algorithme, le modèle $\mathcal{M}c$ représente un ensemble vide puisqu'il va être créé pour la première fois, par la suite il sera mis à jour dynamiquement. Cet algorithme produit à la sortie, le modèle de comportant de l'utilisateur ($\mathcal{M}c$) mis à jour selon les nouvelles observations rajoutées à l'historique.

Dans cet algorithme, nous précisons un paramètre μ qui prend sa valeur dans l'intervalle [0,1]. Le *paramètre* μ représente le seuil au-delà duquel deux observations ne sont pas considérées comme successives selon leur *timestamp*. Il représente la durée limite entre deux observations pour conclure que ces deux derniers ont une certaine liaison ensemble, *i.e.* l'utilisateur a une intention *I1* qui a induit à une intention *I2*.

L'algorithme de classification traite toutes les nouvelles observations (O_S) enregistrées dans l'historique. Cet algorithme se charge, pour chacune de ces observations (O_{Si}), de sélectionner l'observation (O_{Sii}) enregistrée directement après O_{Si} dans l'historique (fonction *GetNextObservation* de la Figure 70). Pour conclure que deux observations représentent deux situations successives et liées, il faut que la durée qui sépare les deux observations soit raisonnable, plus précisément inférieure à une certaine durée μ (cette durée est définie pas le concepteur du système lors de la phase de conception). Nous ne retenons ainsi que les couples d'observations qui sont considérées comme successives. Par la suite, puisque les états du modèle de comportement ($\mathcal{M}c$), représenté sous forme de chaines de Markov, sont représentés par des *clusters*, nous récupérons pour chaque observation O_{Si} et son successeur O_{Sii} les *clusters* auxquels elles appartiennent (fonction *getCluster* de la Figure 70). L'identifiant du *cluster* contenant chacune de ces observations est enregistré avec chaque observations dans la base des traces, après chaque phase de *clustering*.

Ainsi, après avoir traité toutes les nouvelles observations, nous avons une liste avec tous les couples possibles de *clusters* successifs (regroupant les observations traitées et leurs successeurs). Cette liste sera traitée, par la suite, afin de déterminer pour chaque couple la probabilité de transition d'un *cluster* de départ ($\mathcal{CLO}_{Si}$), représentant l'état de départ de $\mathcal{M}c$, vers son *cluster* d'arrivée ($\mathcal{CLO}_{Sii}$), représentant l'état d'arrivée de $\mathcal{M}c$ (fonction *getNewTransition* de la Figure 70). Le calcul de cette probabilité de transition repose sur le nombre de transitions de $\mathcal{CLO}_{Si}$ à $\mathcal{CLO}_{Sii}$ divisé par le nombre de transitions possibles à partir de $\mathcal{CLO}_{Si}$. Cette liste de couples de *clusters* avec leur probabilité de transition va permettre par la suite, de mettre à jour le modèle de comportement de l'utilisateur (fonction *UpdateMarkovChainModel* de la Figure 70). Si le modèle $\mathcal{M}c$ est vide, alors l'algorithme va présenter le modèle pour la première fois. Les *clusters* vont représenter les états du modèle et les probabilités vont représenter les transitions possibles entre les états. Dans le cas contraire, le modèle $\mathcal{M}c$ sera plutôt mis à jour. Ceci consiste à mettre à jour les probabilités de transition avec les nouvelles probabilités calculées, pour les *clusters* déjà enregistrés, et de rajouter également les nouveaux *clusters* avec leur probabilité de transition.

Comme l'illustre la Figure 70, cet algorithme commence par déterminer pour chaque observation O_{Si} si elle a un successeur O_{Sii} dans l'historique, en appliquant la fonction *GetNextObservation* (ligne 6-11). Cet algorithme accède à la base des traces et à partir de l'observation O_{Si} récupère celle qui vient tout juste après (O_{Sii}). Ensuite, afin de déterminer si ces deux observations peuvent être successives, cet algorithme compare le *timestamp* enregistré avec l'observation O_{Si} et celui enregistré avec l'observation (O_{Sii}) (ligne 7). Si la différence entre ces deux *timestamps* dépasse le seuil μ, alors ces deux observations ne sont

pas considérées comme successives. Dans le cas contraire, cet algorithme récupère les identifiants des *clusters* CLO_{Si} et CLO_{Sii} contenant ces observations pour, ensuite, rajouter le couple < CLO_{Si}, CLO_{Sii} > à la liste *MapPossibleSituation* (ligne 10). A partir de la liste *MapPossibleSituation,* l'algorithme va déterminer, à partir de la fonction *getNewTransition,* les probabilités de transition d'un *cluster* (CLO_{Si}), représentant un état dans le modèle de comportement $\mathcal{M}c$ représenté par les chaines de markov, à un autre *cluster* CLO_{Si} (ligne 14). Cette fonction utilise la Formule 8 pour calculer les probabilités. Le résultat sera par la suite enregistré dans la liste *MapNewTransition* (ligne 14). Finalement, en ayant la nouvelle liste des transitions, l'algorithme lance la mise à jour du modèle des chaines de Markov $\mathcal{M}c$. Ceci consiste à mettre à jour les probabilités de transitions ou à rajouter d'autres états au modèle, en se basant sur la fonction *updateMarkovChainModel* (ligne 14).

Le processus de prédiction, décrit dans la section suivante, utilise les résultats du processus de classification afin de prédire le service qui satisfait l'intention suivante de l'utilisateur.

8.2.3. Le processus de prédiction

Un comportement plus proactif peut être obtenu avec la prédiction des besoins futurs des utilisateurs. Le but de ce processus de prédiction est de prédire l'intention future de l'utilisateur afin de lui proposer le prochain service qui peut l'intéresser. De cette manière, l'utilisateur n'aurait pas à le solliciter activement. Ce processus est déclenché lorsqu'un processus de découverte de services est effectué avec succès. Ainsi, sur la base de l'intention courante de l'utilisateur I_U et de son contexte courant Cx_U, le processus de prédiction est capable de proposer le service suivant à celui proposé par la découverte (*cf.* Chapitre 7).

8.2.3.1. Principe de l'algorithme de prédiction de services

Le processus de prédiction de services se base sur le modèle de comportement $\mathcal{M}c$ (*les chaines de markovs*) mis à jours lors de la phase précédente de classification. Le processus de prédiction se charge, ensuite, de chercher l'état (Sti) de $\mathcal{M}c$ qui est le plus proche (*i.e.* similaire sémantiquement) de la situation actuelle de l'utilisateur, caractérisée par sont intention et contexte courant, et d'en déduire la situation suivante, celle qui est la plus probable. Plus précisément, et comme l'illustre la Figure 71, ce processus compare sémantiquement, pour chaque *cluster* représenté comme état dans les chaines de Markov $\mathcal{M}c$, l'intention immédiate de l'utilisateur (I_U) avec l'intention du centroïde du *cluster* (I_{Sti}). Ensuite, il compare sémantiquement le contexte courant de l'utilisateur (Cx_U) avec le contexte du centroïde du *cluster* (Cx_{Sti}). Si le score final (degré de mise en correspondance entre l'intention et le contexte de l'utilisateur et ceux du centroïde du *cluster*) est acceptable (dépasse un certain seuil), alors ce *cluster* est retenu comme un *cluster* candidat. Une fois ce traitement effectué sur l'ensemble des *clusters* dans les chaines de Markov $\mathcal{M}c$, alors le *cluster* ayant eu le score le plus élevé est retenu. Le service futur qui sera offert à l'utilisateur représente le service du centroïde du *cluster* retenu à la fin.

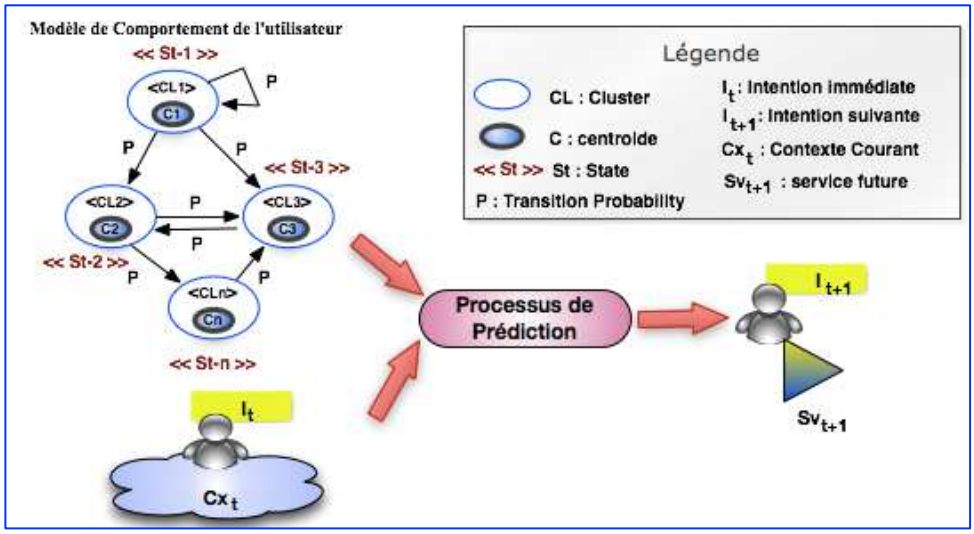


Figure 71. La prédiction de l'intention et du service futurs durant le processus de prédiction

La Figure 72 illustre notre algorithme pour prédire l'intention future de l'utilisateur et par conséquent le prochain service (Sv_{future}) le plus approprié qui peut l'intéresser.

L'algorithme de prédiction prend en entrée une *intention de l'utilisateur* ($I_{\mathcal{U}}$), une description du *contexte courant* de l'utilisateur ($Cx_{\mathcal{U}}$) et le modèle de comportement de l'utilisateur ($\mathcal{M}c$). Il produit, en sortie, le service prochain (Sv_{future}) prédit qui répond à l'intention future de l'utilisateur dans un contexte semblable à son contexte courant et qui peut l'intéresser.

Dans cet algorithme, nous précisons trois paramètres α, β et ρ qui prennent leurs valeurs dans l'intervalle [0,1]. Les paramètres α et β représentent les mêmes paramètres utilisés dans l'algorithme de découverte de services (*cf.* section 7.2.2.2). Le *paramètre* α représente le seuil (le degré de mise en correspondance intentionnelle minimal à accepter) au-delà duquel l'algorithme procède à la mise en correspondance contextuelle. Si le degré de similarité des intentions est supérieur ou égal au seuil α (le degré de mise en correspondance contextuel minimal à accepter), alors l'algorithme lance la mise en correspondance entre les contextes. Le *paramètre* β représente le seuil au-delà duquel l'algorithme procède au calcul du score global de la mise en correspondance. Si le degré de similarité des descriptions de contexte est supérieur ou égal au seuil β, alors l'algorithme considère l'état (Sti) du modèle de comportement ($\mathcal{M}c$) comme un état candidat représentant une situation qui répond à l'intention $I_{\mathcal{U}}$ et au contexte de l'utilisateur $Cx_{\mathcal{U}}$. Le paramètre ρ représente le seuil au-delà duquel une probabilité de transition est acceptable. En d'autres termes, il exprime la probabilité de transition minimale qu'il faut accepter lors de la prédiction du service futur.

Procedure services prediction (I_U, CX_U, $\mathcal{M}c$)

(1) St_{ranked} = Ø /* *Liste des états classés avec leurs scores de mise en correspondance* */

(2) $St_{observed}$ = Ø /* *L'état de* $\mathcal{M}c$ *le plus similaire à l'intention* I_U *et au contexte de l'utilisateur* CX_U */

(3) Sv_{future} = Ø /* *le service future qui sera proposé à l'utilisateur*/

(4) $St_{successor}$ = Ø /* *La liste des états suivants* */

(5) I_{score} = 0 /* *Le score de mise en correspondance intentionnelle* */

(6) CX_{score} = 0 /* *Le score de mise en correspondance contextuelle* */

(7) $Score$ = 0 /* *Le score de mise en correspondance globale* */

(8) I_{Sti} = Ø /* *L'intention du centroide d'un cluster*/

(9) CX_{Sti} = Ø /**Le contexte du centroide d'un cluster*/

(10) **For each** Sti de $\mathcal{M}c$ **Do**

(11) I_{Sti} = *getIntention* (Sti) /* *récupérer l'intention du centroide du cluster* Sti*/

(12) I_{score} = *IntentionMatching* (I_U, I_{Sti} , *OntoI*, *OntoV*)

(13) **If** $I_{score} > \alpha$ **Then**

(14) CX_{Sti} = *getContext* (Sti) /* *récupérer le contexte du centroide du cluster* Sti*/

(15) C_{score} = *ContextMatching* (CX_U, CX_{Sti}, *OntoCX*)

(16) **If** $C_{score} > \beta$ **Then**

(17) $Score = (I_{score} + C_{score}) / 2$

(18) $St_{ranked} = St_{ranked} \cup \{< Sti , Score >\}$ /* Rajouter ce *cluster* à la liste des candidats*/

(19) **End If**

(20) **End If**

(21) **End For**

(22) **If** St_{ranked} not empty **Then**

(23) $St_{observed}$ = *gestBestState* (St_{ranked}) /* *récupérer le cluster ayant obtenu le score le plus élevé de mise en correspondance* */

(24) Sv_{future} = *getNextProbableService* ($St_{observed}$) /* *récupérer le service future à partir du centroide du cluster suivant le plus probable à partir du cluster* $St_{observed}$ */

(25) **End If**

(26) **Return** Sv_{future}

(27) **End procedure**

Figure 72. Algorithme de prédiction de services basée sur l'intention et le contexte

Pour chaque état (représenté par le centroïde d'un *cluster*) du modèle de comportement ($\mathcal{M}c$), l'algorithme de prédiction commence par récupérer l'intention du centroïde du *cluster* (I_{Sti}) (ligne 11) et lance par la suite une mise en correspondance intentionnelle avec l'intention de l'utilisateur ($I_{\mathcal{U}}$) (ligne 12). Cette mise en correspondance représente celle utilisée dans le processus de découverte de services, présentée à la section 7.2.2.3. Afin d'évaluer le score retourné, nous utilisons le seuil α fixé au préalable par le concepteur. Ce seuil représente le score minimum au-delà duquel deux intentions sont considérées comme similaires. Ceci permet d'améliorer la performance de l'algorithme en ne continuant le traitement de la mise en correspondance contextuelle que dans le cas où les intentions correspondent sémantiquement. Ensuite, dans le cas où le score de la mise en correspondance intentionnelle dépasse le seuil α (ligne 13), l'algorithme continue par la récupération du contexte du centroïde du *cluster* (Cx_{Sti}) (ligne 14) et lance par la suite une mise en correspondance contextuelle avec le contexte de l'utilisateur ($Cx_{\mathcal{U}}$) (ligne 15). Pour comparer deux descriptions de contexte (Cx), nous utilisons également *une mise en correspondance contextuelle* qui détermine la similarité entre les observations des descriptions contextuelles. Cette mise en correspondance contextuelle est celle utilisée dans l'algorithme de *clustering*, présenté dans la section 8.2.2.1. Par la suite, pour les mêmes rasions justifiant l'usage d'un seuil α dans la mise en correspondance intentionnelle, cet algorithme évalue le score retourné par la mise en correspondance contextuelle, en se référant au seuil β fixé au préalable par le concepteur (ligne 16). Ce seuil représente le score minimum au-delà duquel deux contextes sont considérés comme similaires. Dans le cas où le score de la mise en correspondance contextuelle dépasse le seuil β, l'algorithme procède au calcul du score de mise en correspondance final (ligne 17). Ce *cluster* traité représente, dans ce cas, un *cluster* candidat qui pourrait représenter le mieux la situation actuelle de l'utilisateur (ligne 18). Dans le cas contraire, ce *cluster* sera éliminé.

Après avoir traité tous les *clusters* appartenant au modèle $\mathcal{M}c$, si au moins un des *clusters* est retenu comme *cluster* candidat (ligne 22), alors l'algorithme procède à la récupération du *cluster* suivant et à la vérification de sa probabilité de transition qui va déterminer si le *cluster* suivant peut être retenu. Ainsi, l'algorithme commence par récupérer, parmi les *clusters* appartenant à la liste des candidats (ligne 23), celui ayant le score le plus élevé. Ensuite, à partir du modèle de comportement $\mathcal{M}c$ et du *cluster* candidat sélectionné ($St_{observed}$), l'algorithme va (i) récupérer tous les *clusters* suivants qui sont probables et (ii) déterminer celui ayant une probabilité de transition la plus élevée et qui dépasse un certain seuil de prédiction ρ (ligne 24). Si aucun *cluster* n'est trouvé, alors l'algorithme s'arrête avec aucune prédiction possible du prochain service qui peut intéresser l'utilisateur. Sinon, parmi les *clusters*, l'algorithme récupère celui qui a la probabilité de transition la plus élevée et qui dépasse le seuil ρ. Ce *cluster* représentera, si ces conditions sont satisfaites, le *cluster* suivant le plus probable à partir du *cluster* de départ. Le service future (Sv_{future}) à proposer à l'utilisateur est le service composant le centroïde du *cluster* suivant retenu à la fin (ligne 26).

La problématique rencontrée dans cet algorithme de prédiction est *que de nouveaux services peuvent se rajouter aux répertoires sémantiques de services, et qui pourraient*

répondre mieux aux intentions futures de l'utilisateur que ceux déjà enregistrés dans le modèle de comportement. La première option pour résoudre cette problématique est de relancer la découverte de services (cf. section 7.2.2.2) sur l'ensemble des services disponibles en se basant sur l'intention et le contexte du *cluster* suivant retenu. L'inconvénient de cette méthode, compte tenu des résultats que nous avons obtenus lors de l'évaluation de la performance de l'algorithme de découverte de services (cf. section 7.3.4), est qu'elle va avoir un coût non négligeable en termes de performance. Effectivement, en analysant les résultats obtenus par le processus de découverte, un tel choix risque de doubler le temps de réponse de notre algorithme de prédiction. Ainsi, pour des raisons de performance nous avons choisi de ne pas relancer le mécanisme de découverte et de proposer le service déjà enregistré dans le modèle de comportement.

8.2.3.2. Complexité de l'algorithme de prédiction de services

Afin de déterminer la complexité de l'algorithme de prédiction, nous l'avons reformulé comme suit :

Procédure Prédiction de Services

Initialisation des variables St_{ranked}, $St_{observed}$, Sv_{future}, $St_{successor}$, *Iscore,* Cx_{score}, *Score,* I_{Sti} et Cx_{Sti}
Début
Pour chaque *ensemble S1 (boucle 1)*
    Calculer la mise en correspondance intentionnelle (Iscore)
    Si *Iscore* > α
        Pour chaque *ensemble S2 (boucle 2)*
            Pour chaque *ensemble S3 (boucle 3)*
                Calculer la mise en correspondance contextuelle (Cscore)
            Fin pour
        Fin pour
        Si *Cscore* > β
                Calculer le score final de mise en correspondance (Score)
        Fin Si
    Fin Si
Fin pour
Retourner le service futur le plus probable
Fin

De même que la complexité des algorithmes de découverte de services (*cf.* section 7.2.2.5) et de clustering (*cf.* section 8.2.2.1.2), l'algorithme de prédiction de services est un algorithme a complexité *polynomiale de degré trois.* Cette algorithme se base essentiellement sur une première boucle (*boucle 1*) qui est imbriquée. La deuxième boucle (*boucle 2*) représente également une boucle imbriquée. Dans cet algorithme, la première boucle (boucle 1) s'exécutent sur l'ensemble des états stockées dans le modèle de comportement de l'utilisateur

dans la base de données (de taille N). Ensuite, la boucle 2 s'exécutent sur l'ensemble des observations contextuelles décrites dans le contexte de l'état (le centroïde du *cluster* qui le représente) (de taille M). La boucle 3, quant à elle, s'exécutent sur l'ensemble des observations contextuelles décrites dans le contexte du courant de l'utilisateur (de taille L).

Ainsi, lors de l'exécution de l'algorithme de prédiction, la *boucle 1* s'exécute N fois. A chaque exécution de la *boucle 1* sur un état du modèle de comportement, la *boucle 2* s'exécute M fois. Ensuite, à chaque fois que la *boucle 2* s'exécute sur une observation contextuelle de l'état, la *boucle 3* s'exécute L fois. Par conséquent, le traitement de la boucle 3 s'exécutent au total *N*M*L* fois. Ainsi, nous déterminons que la complexité de notre algorithme est *O(N*M*L)*. Toutefois, au pire des cas où les trois boucles s'exécutent N fois (avec N le nombre maximum entre N, M et L), la complexité totale pour les trois boucles est $O(N^3)$. Ainsi, l'algorithme de prédiction de services guidé par l'intention et le contexte que nous proposons est un algorithme de complexité *polynomiale de degré trois*.

8.3. Implementation et Evaluation

8.3.1. Implémentation

Comme la découverte de services, le mécanisme de prédiction de services, que nous présentons dans ce chapitre, a été implémenté en langage Java. Cette implémentation comporte trois parties, à savoir, l'implémentation de l'algorithme de *clustering*, l'implémentation de l'algorithme de *classification* et l'implémentation de l'algorithme de *prédiction*. Ces implémentations se basent sur la définition de trois interfaces, à savoir l'interface de *clustering*, l'interface de *classification* et l'interface de *prédiction*. L'usage d'interfaces Java fournit à l'implémentation un niveau supplémentaire d'abstraction, garantissant une architecture flexible. Pour avoir une implémentation qui puisse évoluer, nous cachons les implémentations de nos algorithmes derrière les interfaces. Dans ce cadre, une interface est vue comme un *contrat*, et les composants offrant l'implémentation de ce contrat doivent le respecter. Cette stratégie nous permet de remplacer facilement nos algorithmes, grâce notamment à un fichier de configuration. Ce fichier mentionne les implémentations à lancer lors de l'exécution. Ainsi, de nouveaux algorithmes peuvent être développés et testés par la suite. Les implémentations des interfaces utilisent ainsi le patron de conception « *strategy pattern* » pour fournir un changement souple de stratégie dans la perspective d'assurer un caractère flexible, réutilisable et échangeable de notre architecture des SIP. Ainsi, pour charger la bonne stratégie, nous utilisons un fichier de configuration (*config.properties*) où nous attribuons le composant de la stratégie à utiliser. Au cours du démarrage, ce composant défini au préalable est chargé comme étant la stratégie par défaut.

Il est à noter que les traces, les *clusters* reconnus ainsi que le modèle de comportement sont maintenus dans trois tables différentes d'une base de données, à savoir la table *traces* pour enregistrer l'ensemble des observations, la table *cluster* pour sauvegarder les *clusters* reconnus et la table *markovchain* pour maintenir dynamiquement le modèle de comportement de l'utilisateur. Cette base de données est organisée en fonction de l'utilisateur, offrant ainsi

des données individualisées. L'avantage d'un tel choix est la *personnalisation des données*. Par contre, l'inconvénient est que le processus de prédiction sera moins pertinent au départ puisqu'il ne va pas y avoir assez de traces pour pouvoir faire des prédictions plus précises.

Premièrement, le composant implémentant l'*interface clustering* reçoit, comme entrée, l'observation enregistrée dans la trace de la base de données et lance, par la suite, l'algorithme de *clustering* (*cf.* section 8.2.2.1), lequel retourne enfin la mise à jour des *clusters* déjà reconnus ou la création d'un nouveau *cluster* dans la table *cluster* de la base de données. Cet algorithme permet également de mettre à jour la table *traces* en rajoutant, pour chaque observation, l'identifiant du *cluster* auquel elle appartient. Deuxièmement, l'implémentation de l'*interface de classification* reçoit comme entrée l'ensemble des *clusters* et des observations enregistrés dans la base de données. Elle lance l'algorithme de Markov Chain (*cf.* section 8.2.2.2) et retourne le modèle de comportement de l'utilisateur mis à jour dynamiquement dans la base de données. Enfin, le composant implémentant l'*interface de prédiction* reçoit comme entrée, la situation courante de l'utilisateur (son intention et son contexte courant) et son modèle de comportement qu'elle récupère de la base de données. Elle implémente l'algorithme de prédiction décrit dans la section précédente (*cf.* section 8.2.3), lequel se charge de retourner l'intention suivante la plus probable et le service le plus approprié qui peut intéresser l'utilisateur.

Nous détaillons, dans les sections suivantes, chacune de nos implémentations des algorithmes de *clustering*, de *classification* et de *prédiction*. Nous nous basons sur le même principe de structuration du code de développement que dans l'implémentation de l'algorithme de découverte de services, en utilisant le patron de conception « *strategy pattern* » pour fournir un changement souple de stratégie. L'implémentation de la partie *persistance* représente également celle utilisée pour manipuler les ontologies et les descriptions de services dans le processus de découverte de services (cf. section 7.3.2.2).

8.3.1.1. Implémentation de l'algorithme de clustering

L'implémentation du mécanisme de *clustering* est organisée, essentiellement, autour de l'implémentation de trois interfaces distinctes, comme l'illustre la Figure 73 :

- L'interface « ***ILearningFacade*** » représente le point d'entrée de ce mécanisme de *clustering* et offre un ensemble de méthodes supportant la gestion des ontologies ainsi que le *clustering* des situations de l'utilisateur enregistrées dans l'historique, telles que les méthodes *addontologies*, *removeontologies* et *getCluster* ;
- L'interface « ***IPersistenceManager »*** agit comme une façade entre le composant *« LearningFacadeImpl »* et le répertoire d'ontologies, ce qui permet l'accès et le chargement des ontologies (L'implémentation de cette interface est la même que celle utilisée par la découverte, *cf.* la section 7.3.2.2) ;
- L'interface « ***IClusteringEngine »*** est responsable du *clustering* des situations de l'utilisateur, qui sont composées de son intention, son contexte et le service sélectionné à un instant donné. Elle utilise une interface « *IClusteringFacade* » afin de

donner plus de flexibilité à notre implémentation. Elle permet d'étendre facilement le mécanisme de *clustering* des situations de l'utilisateur par l'ajout de nouveaux algorithmes de *clustering*.

Dans l'architecture (*cf.* Chapitre 9), l'interface « *IClusteringEngine* » fournit la méthode *loadDefaultCluster* laquelle permet de déterminer la stratégie à mettre en place. Elle fournit également une méthode principale de *clustering* des situations de l'utilisateur, appelée *getCluster*. Cette méthode accède à la base de données et prend en entrée, une *nouvelle observation enregistrée dans les traces* et l'ensemble des *clusters déjà reconnus*. Elle fait appel, par la suite, à la bonne méthode de *clustering*, selon la stratégie spécifiée au préalable dans le fichier de configuration, laquelle doit découvrir pour chaque observation à quel *cluster* elle va appartenir, ou si elle va faire l'objet de création d'un nouveau *cluster*. Elle offre en sortie la mise à jour des *clusters* déjà reconnus dans la base de données.

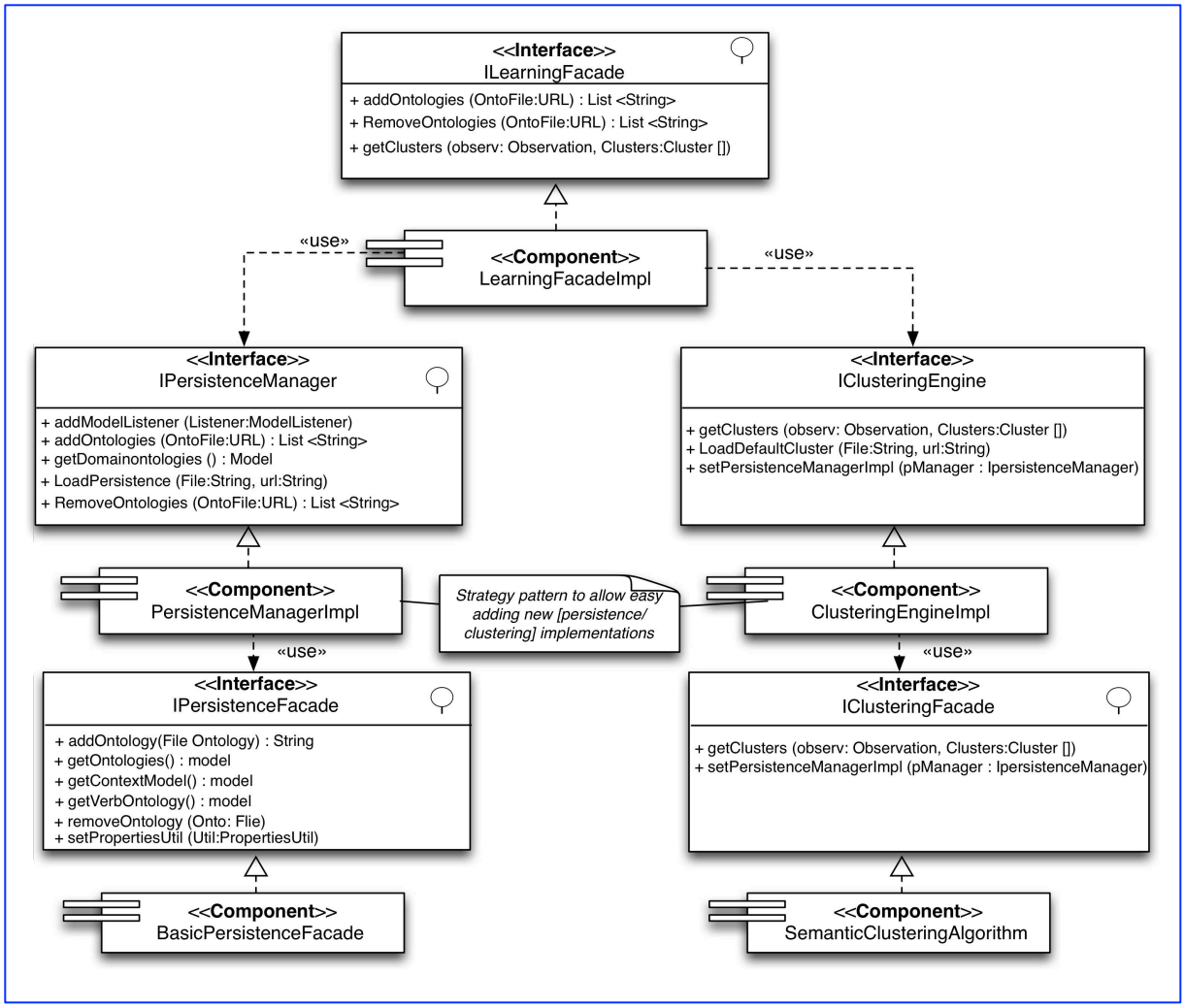


Figure 73. Les composants implémentant le mécanisme de clustering

Le composant « *ClusteringEngineImpl* » implémente l'interface « *IClusteringEngine* ». Ceci revient à implémenter la méthode *getCluster* qui va déterminer et mettre à jour un

ensemble de *clusters* regroupant les observations similaires de l'utilisateur. Les observations sont enregistrées dans la base de données sous forme de *timestamp*, *intention*, *contexte*, *service, identifiant du cluster*. Quant aux *clusters*, ils sont stockés dans une table séparée sous la forme de l'*intention,* le *contexte* et le *service attribué au centroïde* du *cluster*. Cette phase de *clustering* représente un processus en arrière plan qui va mettre à jour la base de données, plus précisément la table *cluster,* en mettant à jour les *clusters* existants ou en créant de nouveaux *clusters*, et la table *traces,* à laquelle il va rajouter pour chaque nouvelle observation l'identifiant du *cluster* auquel elle appartient.

Afin de permettre l'évolution de l'API, cette première implémentation « *ClusteringEngineImpl* » utilise l'interface « *IClusteringFacade* », laquelle offre une interface générale pour l'implémentation de nouvelles méthodes de *clustering*. La sélection de l'implémentation de l'algorithme de *clustering* à déployer se fait ainsi par la simple édition d'un fichier de propriétés.

Dans le cadre de nos travaux, nous mettons en œuvre l'algorithme de *clustering*, proposé dans la section 8.2.2.1, à travers le composant « *SemanticClusteringAlgorithm* » qui se charge de regrouper sémantiquement les situations des utilisateurs selon l'intention et le contexte. Pour chaque observation de l'utilisateur enregistrée dans la table *traces* de la base de données, après une découverte de services, le « *SemanticClusteringAlgorithm »* permet de calculer un score de mise en correspondance entre cette observation et les centroïdes des *clusters* existants, et de mettre à jour ces *clusters* existants ou d'en créer un nouveau. Cette implémentation fait appel, à l'instar de l'implémentation de l'algorithme de découverte (*cf.* section 7.3.2.2) de services, à une implémentation des classes « *ContextMatching* » et « *IntentionMatching* ». Le « *ContextMatching* » est responsable de déterminer le score de la mise en correspondance contextuelle. Le « *IntentionMatching* » est responsable de déterminer le score de la mise en correspondance intentionnelle.

8.3.1.2. Implémentation de l'algorithme de classification

L'implémentation du mécanisme de *classification* est organisée, essentiellement, autour de deux interfaces, comme l'illustre la Figure 74 :

- L'interface « ***ILearningFacade***» représente le point d'entrée de ce mécanisme de *classification* et offre la méthode *doClassification* supportant la classification des *clusters*, déterminés lors de la phase de *clustering*, afin de maintenir à jour le modèle de comportement de l'utilisateur dans la base de données ;
- L'interface « ***IClassificationEngine »*** est responsable de la *classification* des *clusters* reconnus. Elle utilise une interface « *IClassificationFacade* » afin de donner plus de flexibilité à notre implémentation. Elle permet d'étendre facilement le mécanisme de *classification* des *clusters* par l'ajout de nouveaux algorithmes de *classification*.

Comme l'illustre la Figure 74, l'interface « *IClassificationEngine* » fournit la méthode *loadDefaultClassifier* laquelle permet de déterminer la stratégie à mettre en place, à partir du

fichier de configuration. Elle fournit également une méthode principale de classification des *clusters* déjà reconnus, appelée *doClassification*. Cette méthode accède à la base de données et prend en entrée, l'ensemble des *clusters déjà reconnus* et l'ensemble *d'observations*. Elle fait appel par la suite à la bonne méthode de classification, selon la stratégie spécifiée au préalable dans le fichier de configuration, laquelle doit mettre à jour le modèle de comportement de l'utilisateur (sous forme de chaines de Markov). Elle offre en sortie la mise à jour des chaines de Markov dans la base de données.

Le composant « *ClassificationEngineImpl* » implémente l'interface « *IClassificationEngine* », laquelle fournit la méthode de classification des *clusters* reconnus selon les observations de l'utilisateur, appelé *doClassification*. Chaque méthode doit être capable de déterminer et mettre à jour dynamiquement le modèle de comportement de l'utilisateur en se basant sur l'ensemble des *clusters* déterminés et sur les observations enregistrées dans la base de données. Le composant « *ClassificationEngineImpl* » implémente l'interface « *IClassificationFacade* », laquelle lui offre une interface générale pour l'implémentation de nouvelles méthodes de *classification*. Ce composant implémente la méthode *doClassification* qui met à jour le modèle de comportement de l'utilisateur. Le modèle de comportement est enregistré dans la table « *MarkovChain* » de la base de données. Il contient des informations sur le *cluster de départ*, *le cluster d'arrivée*, *le nombre de transitions qui existent entre ces deux clusters selon les observations enregistrées dans les traces*, *le nombre de transition de ce cluster de départ vers n'importe quel autre cluster* et finalement la *probabilité de transition* calculé selon ces deux dernières informations.

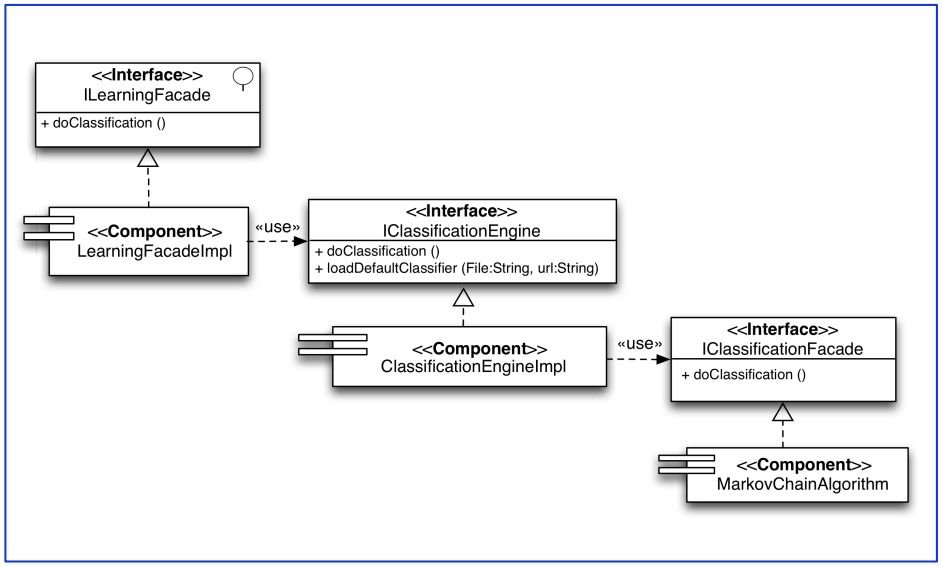


Figure 74. Les composants implémentant le mécanisme de classification

Le mécanisme de classification, proposé dans la section 8.2.2.2, est mis en œuvre à travers le composant « *MarkovChainAlgorithm* » qui se charge de déterminer et maintenir à jour le modèle de comportement de l'utilisateur. Pour chaque *cluster* reconnu lors de la phase de *clustering* et enregistré dans la base de données, le « *MarkovChainAlgorithm* » permet de calculer sa probabilité de transition vers un autre *cluster*, en se basant sur l'ensemble des

observations enregistrées dans l'historique. Après le traitement de tous les *clusters*, ce composant fournit en sortie un modèle de comportement de l'utilisateur mis à jour. Ce modèle est maintenu dans la table *marvchain* de la base de données.

8.3.1.3. Implémentation de l'algorithme de prédiction

En suivant la même structure de développement que les algorithmes de découverte de services et de *clustering*, l'implémentation du processus de *prédiction* est organisée autour de trois implémentations d'interfaces, comme l'illustre la Figure 75 :

- L'interface « ***IPredictionFacade »*** représente le point d'entrée de ce processus de *prédiction* et offre un ensemble de méthodes qui supportent la gestion des ontologies ainsi que la prédiction de l'intention future de l'utilisateur et du service qui répond à cette intention, tels que les méthodes *addontologies* et *doPrediction* ;
- L'interface « ***IPersistenceManager »*** agit comme une façade entre le composant *« PredictionFacadeImpl »* et le répertoire d'ontologies, ce qui permet l'accès et le chargement des ontologies (L'implémentation de cette interface est la même que celle utilisée par la découverte, *cf.* la section 7.3.2.2) ;
- L'interface « ***IPredictionEngine »*** est responsable de la prédiction de l'intention future de l'utilisateur et du service le plus approprié. Elle utilise une interface « *IPredictionMatcherFacade* » afin de permettre d'étendre facilement le mécanisme de prédiction par l'ajout de nouveaux algorithmes.

Dans la Figure 75, l'interface « *IPredictionEngine* » fournit la méthode *loadDefaultPrediction* laquelle permet de déterminer la stratégie à mettre en place, à partir du fichier de configuration. Elle fournit également une méthode principale de prédiction de service, appelée *doPrediction*. Cette méthode reçoit comme entrée l'intention de l'utilisateur enrichie par son contexte courant de l'utilisateur. Elle fait appel par la suite à la bonne méthode de prédiction, selon la stratégie spécifiée au préalable dans le fichier de configuration, laquelle doit prédire, selon le modèle de comportement enregistré dans la base de données, l'intention future de l'utilisateur ainsi que le service suivant. Elle offre en sortie le service qui peut intéresser l'utilisateur.

Le composant « *PredictionEngineImpl* » représente l'implémentation de l'interface « *IPredictionEngine* », laquelle fournit la méthode de prédiction de l'intention et du service future de l'utilisateur, appelé *doPrediction*. L'implémentation du composant « *PredictionEngineImpl* » utilise l'interface « *IPredictionMatcherFacade* », laquelle offre une interface générale pour l'implémentation de nouvelles méthodes de *prédiction*.

Nous mettons en œuvre le mécanisme de prédiction de services proposé dans cette thèse par l'implémentation du « *ContextIntentionPredictionMatch* » lequel se charge de découvrir et sélectionner la situation future de l'utilisateur, à partir du modèle de comportement et de son intention et contexte courant. Pour chaque état du modèle de comportement, le *ContextIntentionPredictionMatch* permet de calculer un score de mise en correspondance

entre les identifiants des états, représentés par les centroïde des *clusters*, et l'intention et le contexte courant de l'utilisateur. Cette classe, comme le « *SemanticClusteringAlgorithm* » *(cf.* 3.1.1) et similairement à la découverte de services (*cf.* section 7.3.2.2), fait appel à deux classes, à savoir la classe « *ContextMatching* » pour déterminer le score de la mise en correspondance contextuelle et la classe « *IntentionMatching* » pour déterminer le score de la mise en correspondance intentionnelle.

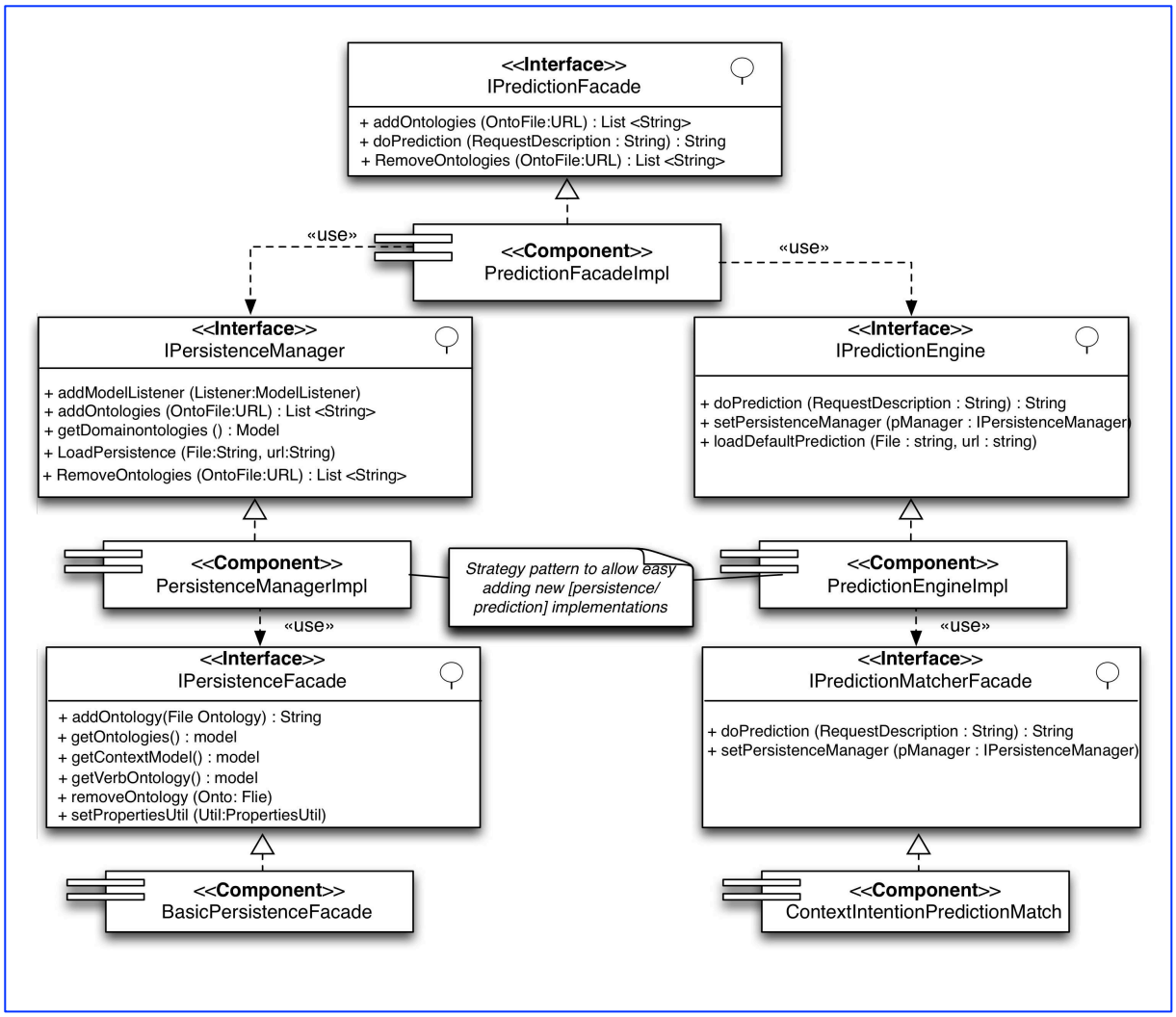


Figure 75. Les composants implémentant le mécanisme de prédiction de services

La section suivante analyse les résultats des expérimentations qui ont été menées afin d'évaluer le processus d'apprentissage et de prédiction guidé par l'intention et le contexte.

8.3.2. Evaluation

Dans le cadre de nos expérimentations, nous avons déployé nos algorithmes de *clustering*, de *classification* et de *prédiction* sur une machine de processeur *Intel Core i5 1,3 GHz* avec une mémoire de 4 Go. Lors de cette évaluation du processus de prédiction de services, nous nous sommes focalisé essentiellement sur l'évaluation de l'algorithme de *prédiction de*

services. Les algorithmes de *clustering* et de *classification* représentent les deux phases assurant le fonctionnement de l'algorithme de prédiction de services exploitant un modèle de comportement de l'utilisateur inféré des traces. Ces deux phases de *clustering* et de *classification*, ne représentent pas un but final en soi, mais deux étapes nécessaires pour aller vers notre but final, à savoir la prédiction de services. Ainsi, nous avons décidé d'évaluer la globalité de notre approche de prédiction par l'évaluation de l'algorithme de prédiction de services (qui repose sur les résultats des autres algorithmes) et de la qualité du résultat qu'il retourne.

De plus, l'évaluation de ces différents algorithmes a été effectuée sur le même répertoire de services utilisé pour l'évaluation de l'algorithme de découverte de services (*cf.* section 7.3.3). Au cours de cette évaluation, nous avons choisi de ne pas évaluer nos algorithmes sur les machines du Grid'5000. Lors de l'évaluation de l'algorithme de découverte de services, nous avons observé que quelque soit la machine et la configuration utilisées, la performance de notre algorithme reste à peu près constante. Ainsi, nous avons décidé de tester notre algorithme de prédiction que sur une seule architecture.

Afin d'évaluer l'algorithme de prédiction de services, nous avons construit une base de données nommée *Prediction.bd*. Cette base de données contient trois tables : une contenant l'historique de l'utilisateur (*traces*), une gardant les *clusters* reconnus (*clusters*) et une troisième stockant le modèle de comportement de l'utilisateur (*markovChain)*. Nous avons commencé par alimenter cette base de données par un ensemble d'observations enregistrées comme de traces. Ces traces sont décrites par un ensemble d'intentions. Nous rajoutons à ces intentions, différentes descriptions contextuelles et l'identifiant du service qui permet de répondre à cette intention dans ce contexte d'usage. Nous avons défini dix descriptions de contexte, que nous avons affectées arbitrairement aux intentions définies. Les traces comprennent plus de 30 intentions décrites dans différents contextes et qui sont éparpillées dans le temps. Pour la construction de ces traces, nous avons suivi au début un scénario représentant un comportement bien défini de l'utilisateur. Par exemple, l'utilisateur commence par chercher une destination. Ensuite, il réserve un hôtel. Puis, il loue une voiture et demande une carte de sa destination. Finalement, il cherche les musées les plus intéressants dans la destination qu'il a choisi. Les traces suivantes sont définies de manière aléatoire et intégrées à la base de données. Il est à noter que les trace ne sont pas réelles et ont été crées d'une manière fictive. Ceci est dû à la difficulté de convaincre les entreprises de nous fournir des données réelles, pour des raisons de respect de la vie privée.

Par la suite, afin d'évaluer l'algorithme de prédiction de services, nous avons besoin du modèle de comportement de l'utilisateur ainsi que l'ensemble des *clusters* qui regroupe les traces stockées dans la base de données. Pour ce faire, nous commençons par lancer l'algorithme de *clustering* sur l'ensemble des traces. Le résultat de cette étape est l'alimentation de la table *clusters* par un ensemble de *clusters* reconnus. Par la suite, nous exécutons notre algorithme de *classification* sur l'ensemble des traces et des *clusters*. Cette étape nous permet de créer et de mettre à jour le modèle de comportement de l'utilisateur qui sera stocké dans la table *markovchain* de notre base de données. Nous obtenons au final, une

base de données alimentée part un ensemble de traces de l'utilisateur, des *clusters* ainsi que son modèle de comportement. Cette base est ensuite utilisée pour évaluer l'algorithme de prédiction de services.

Comme pour l'évaluation de l'algorithme de découverte de services (*cf.* section 7.3.3), nous avons réalisé un ensemble de mesures, afin d'évaluer la validité des algorithmes proposés. Nous évaluons nos algorithmes en nous basant sur les deux observations principales qui se dégagent de ces expériences, à savoir *le passage à l'échelle* et la *qualité du résultat* obtenu. Pour supporter notre nouvelle vision des SI, il est essentiel de vérifier la faisabilité de notre approche avec un nombre considérable de traces à gérer. Ce problème de passage à l'échelle a mis l'accent sur la *performance* de nos algorithmes d'apprentissage et de prédiction. Nous analysons cet aspect en mesurant l'impact du nombre de traces, de *clusters* et d'états gérés sur la *performance* des algorithmes en termes de temps moyen d'exécution. De plus, à part la performance de nos algorithmes, la qualité des résultats obtenus par l'algorithme de prédiction de services représente un critère très important à prendre en considération et surtout dans le cadre de notre vision centrée utilisateur des SIP, dans laquelle nous devons être précis dans le choix du service que nous proposons à l'utilisateur. Nous analysons ainsi cet aspect en s'inspirant de la mesure de *précision* utilisée dans l'évaluation de la qualité des résultats de l'algorithme de découverte de services (*cf.* section 7.3.4.2). Cette mesure nous permet de vérifier si le service prédit par l'algorithme de prédiction de services, afin de répondre à une intention future de l'utilisateur, est bien celui qui est le plus probable, c.-à-d. si la prédiction est correcte.

Finalement, afin d'évaluer le passage à l'échelle et les qualités des résultats, nous avons formulé neuf requêtes différentes relatives au domaine du voyage et qui doivent permettre à l'algorithme de prédiction de services de prédire les services suivants qui répondent à une intention future de l'utilisateur dans un contexte similaire. Ces requêtes sont indiquées dans le Tableau 7.

Tableau 7. Description des requêtes de l'utilisateurs utilisées dans l'évaluation de la prédiction de services

Requête	Intention	Contexte Courant
Req 1	Rent Car	-Profile.Age = 34, -Profile.Expertise = High -DateTime.Season= Spring, -Location.City = France -Device.Mobile_device = Ipad2, -Resource.Network = Mobile-Connection, -Resource.Memory = 2048, -Resource.Screen = 9,7
Req 2	Search BedAndBreakfast	-Profile.Age = 21, -DateTime.Season= winter -DateTime.Time= morning, -Location.City = France -Device.Mobile_device = Mackbook, -Resource.Screen = 16 -Resource.Network = Ethernet, -Resource.Memory = 2048,
Req 3	Book-up Safari Lodge	-DateTime.Season = Summer, -Profile.Age = 23 -DateTime.Time = Evening, -Location.City = Tanzania -Resource.Network = Wifi, -Device = Intel Core 2 duo -Resource.Memory = 1024, -Resource.Screen =15
Req 4	Get Accommodation	- Profile.Age = 23, - Resource.Battery = Low - Ressource.Microphone = On, - Location.name = home

	Rating	-Location.City = France, - DateTime.Date = 23/11/2010 - DateTime.Season = Summer, - Device = NoteBook - Resource.Network = 4G
Req 5	Locate Five Star European Hotel	-Location.City = Germany, - Profile.Age = 29 -DateTime.Season = winter, - DateTime.Time = night -Profile.Expertise = High, - Profile.Role = Engineer -Resource.Network = Ethernet, - Devices = Intel Core 2 Duo, -Resource.Memory = 2048, - Resource.screen = 16
Req 6	Search Destination	-Location.City = Hawai, - DateTime.Time = Night -Profile.Age = 18, - DateTime.Season = summer -Resource.Network = 4G, - Devices = Samsung Galaxy -Resource.Memory = 562, - Resource.screen = 9 -Profile.Expertise.SurfingExpertise = High -Profile.Expertise.HikingExpertise = Low
Req 7	Locate Sport Organization Destination	-DateTime.Season = Summer, -Profile.Age = 24, -Device = Ipad 2 -DateTime.Time = Morning, -Profile.Role = Student -Profile.Expertise.SurfingExpertise = High, -Resource.Screen = 9 -Location.City = Hawai, -Location.Country = USA -Resource.Network = 3G, -Resource.Memory = 32
Req 8	Get Road Way To	-DateTime.Season = Summer, -Profile.Age = 24 -DateTime.Time = Morning, -Profile.Role = Student -Profile.Expertise.SurfingExpertise = High, -Resource.Screen = 9 -Location.City = Hawai, -Location.Country = USA -Resource.Network = 3G, -Device = Ipad 2,-Resource.Memory = 32
Req 9	Reserve Raja Yoga	-DateTime.Season = Summer, -Profile.Age = 34 -DateTime.Time = evening, -Profile.Role = Teacher -Location.City = Rome, -Location.Country = Italie -Resource.Network = Ethernet, -Device = Intel Core 2 duo -Resource.Memory = 2048, -Resource.Screen = 13

Ces requêtes sont formulées de différentes manières, à l'instar de l'évaluation de la découverte de services (*cf.* section 7.3.3). Nous avons formulé ces requêtes selon 3 répartitions différentes. La première répartition considère des *requêtes qui sont très similaires aux centroïdes des clusters qui représentent un état du modèle de comportement*. La ***Req1*** présente une requête très proche au centroïde d'un *cluster* (état), qui est décrite dans le même contexte mais avec une intention légèrement différente (la cible de la requête est plus générique que celle du *cluster*), alors que la requête ***Req2*** est décrite par la même intention mais avec un description de contexte similaire. Ensuite, la deuxième répartition illustre des situations où (i) *les éléments décrivant l'intention et/ou le contexte ne sont pas décrits dans les ontologies alors qu'il existe un cluster qui est similaire à cette requête* (***Req3***) ; et (ii) *les éléments décrivant l'intention et le contexte sont décrits dans les ontologies alors qu'il n'existe pas de cluster qui est similaire à cette requête* (***Req4***). Finalement, la troisième répartition présente l'influence du seuil sur le résultat final de l'algorithme de prédiction de services. Nous présentons dans cette partie des requêtes qui sont dans la limite du seuil (***Req7*** et ***Req8*** *et* ***Req9***), et d'autres qui sont au-delà du seuil (***Req5*** et ***Req6***)

En résumé, nous synthétisons les caractéristiques des requêtes de l'utilisateur utilisées pour l'évaluation de l'algorithme de prédiction, dans le Tableau 8.

Tableau 8. Caractéristiques des requêtes de l'utilisateur pour la prédiction de services

Requête	Concept n'existe pas dans l'ontologie	N'existe pas d'état correspondant	Complexité concept dans l'ontologie (seuil)	Complexité contexte (couplage faible)	Pas de Résultat/Existe un état correspondant
Req 1					
Req 2					
Req 3	✓				✓
Req 4		✓			
Req 5			✓ *(cible/verbe)*		✓
Req 6			✓ *(cible)*	✓	✓
Req 7			*(dans la limite du seuil)*	✓	
Req 8			✓ *(dans la limite du seuil dans certains cas)*	✓	
Req 9			✓ *(dans la limite du seuil dans certains cas)*	✓	

8.3.2.1. Le passage à l'échelle

Dans les expériences que nous avons menées, nous avons testé le passage à l'échelle des algorithmes de *clustering,* de *classification* et de *prédiction.* Le passage à l'échelle a été représenté par le temps moyen d'exécution, lorsqu'on varie le nombre d'observations disponibles dans la table des *traces* de l'utilisateur, le nombre de *clusters* déjà reconnus dans la table de *cluster* et le nombre d'états disponibles dans la table *markov-chain.*

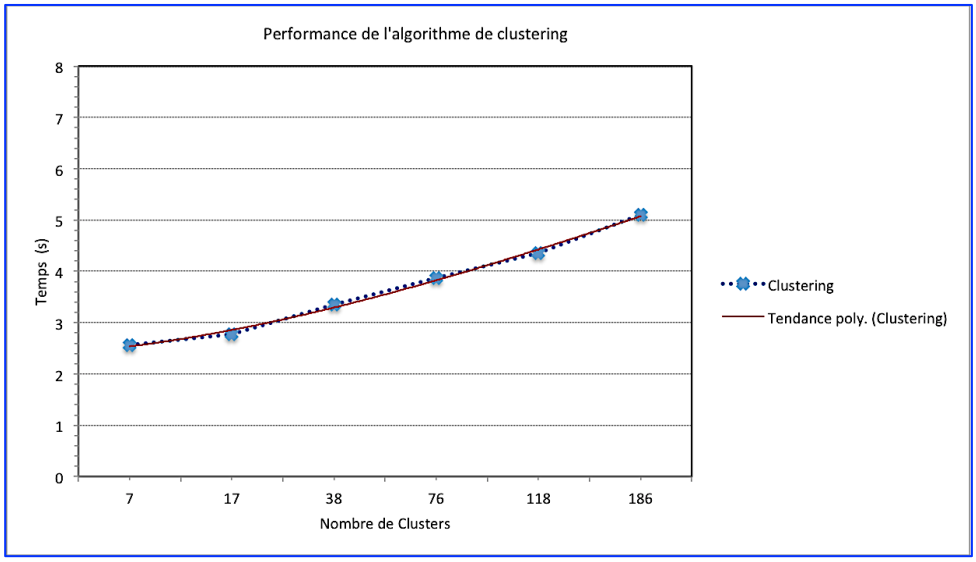


Figure 76. Performance du mécanisme de clustering

Le temps d'exécution de l'algorithme de *clustering* a été mesuré en variant le nombre *clusters* reconnus entre 7 et 186 *clusters*. Ce temps représente le temps moyen d'exécution mis par l'algorithme de *clustering* pour déterminer à quel *cluster* une observation appartient. Comme l'illustre la Figure 76, le temps d'exécution suit une tendance polynomiale de degré trois allant de 2,56 *s* pour 7 *clusters* jusqu'à 5,1 *s* pour 186 *clusters*. Toutefois, même si ce temps s'avère moyen, nous pouvons remarquer que bien que nous ayons augmenté le nombre de *clusters* à traiter de plus de vingt six fois, le temps n'a augmenté que de deux fois. De plus, nous envisageons à court terme d'améliorer ce temps d'exécution en optimisant notre code de développement et en utilisant par exemple les *threads* de Java.

Concernant le temps d'exécution de l'algorithme de *classification,* il a été mesuré en variant le nombre d'observations déjà regroupées en *clusters* dans la base de prédiction entre 10 et 200 observations. Ce temps représente le temps moyen d'exécution mis par l'algorithme de classification afin de mettre à jour dynamiquement le modèle de comportement de l'utilisateur. Comme l'illustre la Figure 77, le temps d'exécution suit une tendance polynomiale allant de 39 *ms* pour 10 observations jusqu'à 398 *ms* pour 200 observations. Toutefois, même si cet algorithme ne prend pas beaucoup de temps pour mettre à jour dynamiquement les chaines de Markov, nous pouvons remarquer qu'il progresse de presque 10 fois en augmentant le nombre d'observations de vingt fois. Cette progression peut être optimisée également en lançant le traitement en parallèle de certaines tâches de l'algorithme.

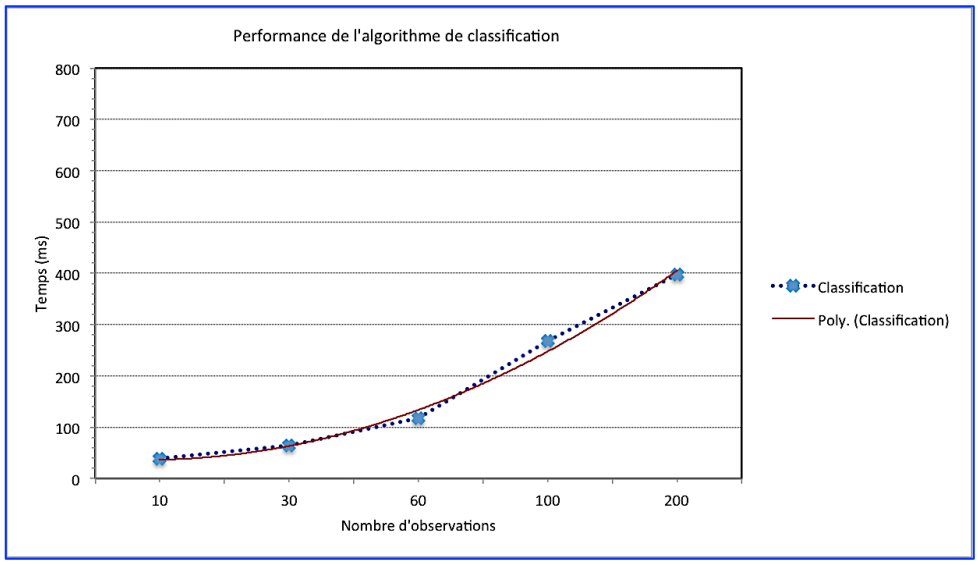


Figure 77. Performance du mécanisme de classification

Finalement, le temps d'exécution de l'algorithme de *prédiction* a été mesuré en variant le nombre d'états dans le modèle de comportement de l'utilisateur stocké dans la base de donnés de prédiction entre 7 et 168 états (lesquels représentent les centroïdes des *clusters*). Ce temps représente le temps moyen d'exécution mis afin de prédire le service suivant qui satisfait une intention future de l'utilisateur selon son intention immédiate et son contexte courant. Comme pour l'algorithme de *clustering*, le temps d'exécution suit une tendance polynomiale de degré trois allant de 1,63 *s* pour 7 états à 4,16 *s* pour 168 états (voir Figure 78). Nous avons

augmenté le nombre d'états plus de vingt cinq fois alors que le temps n'a augmenté que de deux fois et demi. Ceci nous permet de valider la faisabilité du passage à l'échelle de notre algorithme. Toutefois, ces résultats peuvent être optimisés, à l'instar des deux derniers algorithmes de *clustering* et de classification.

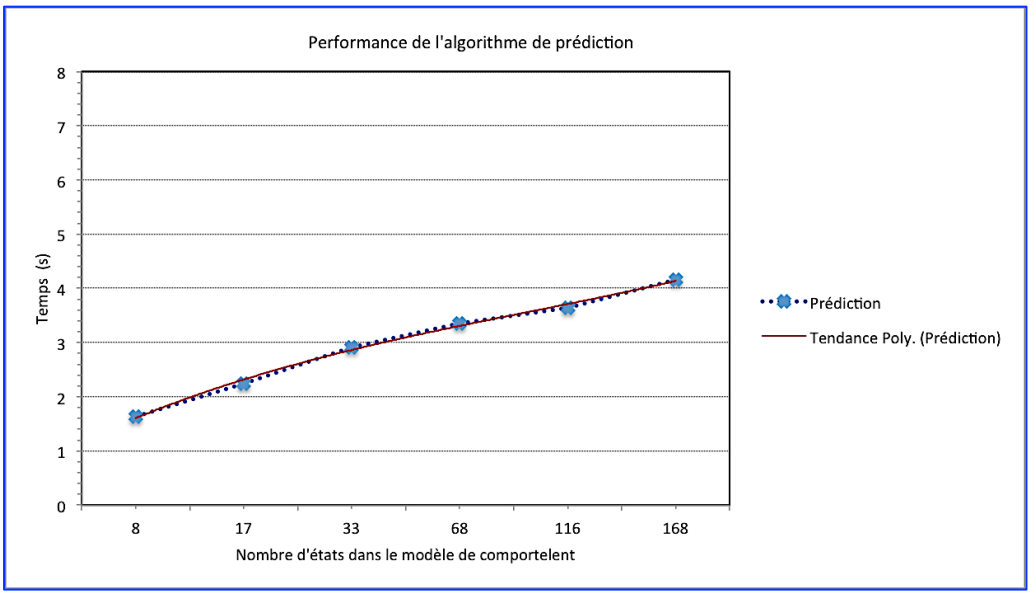


Figure 78. Performance du mécanisme de prédiction

Ces résultats nous permettent de déduire que nos algorithmes assurent un bon passage à l'échelle. Toutefois, la performance de nos algorithmes, surtout pour le *clustering* et la prédiction, est plutôt moyenne. Mais nous restons confiants puisque ces algorithmes peuvent être optimisés afin d'améliorer le temps d'exécution. Ces optimisations représentent une de nos perspectives à court terme.

8.3.2.2. Qualité des résultats

Afin d'évaluer la qualité des résultats, nous nous somme inspiré de la mesure de *précision* utilisée lors de l'évaluation de la qualité des résultats de l'algorithme de découverte de services (*cf.* section 7.3.4.2). Cette mesure est utilisée afin de vérifier si le service prédit par l'algorithme de prédiction services est bien celui qui est attendu.

Concernant l'algorithme de prédiction, nous avons évalué la qualité de l'algorithme de prédiction sur l'ensemble des requêtes formulées au Tableau 7. Nous avons ainsi déterminé au préalable le service que l'algorithme de prédiction de services doit prédire, en se basant également sur le modèle de comportement de l'utilisateur. Nous avons comparé, par la suite, ce service avec le service retourné réellement par l'algorithme. Nous illustrons à la Figure 79 le pourcentage de la qualité obtenu par l'algorithme en variant le nombre d'états dans le modèle de comportement. Ce pourcentage représente la moyenne de la qualité obtenu par chaque requête.

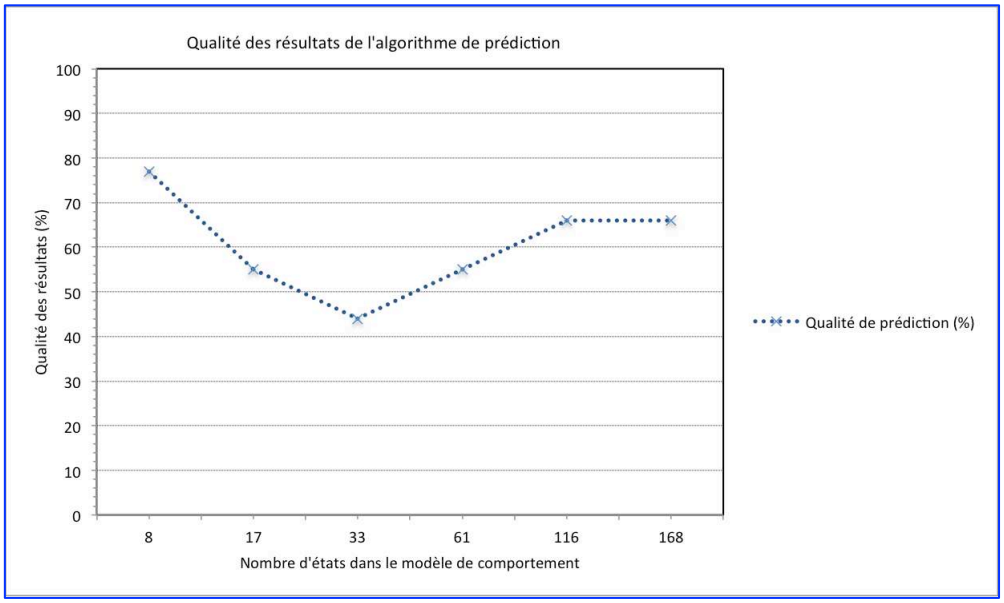


Figure 79. La qualité des résultats du mécanisme de prédiction

Le résultat présenté en Figure 79 indiquent que l'algorithme de prédiction présente une qualité de prédiction plutôt satisfaisante qui tourne autour de 60%. Ces résultats peuvent être expliqués, à l'instar de l'évaluation de la qualité des résultats de l'algorithme de découverte de services (*cf.* section 7.3.4.2), par l'évaluation de certaines situations qui dégradent considérablement la qualité des résultats obtenus. Par exemple, l'évaluation des situations où les éléments de l'intention ne sont pas décrits dans les ontologies alors qu'il existe dans la base de données des *clusters* ou *états* similaires à cette intention dans le contexte de l'utilisateur (*Req3* du Tableau 7). Dans ce cas, l'algorithme de prédiction ne retourne aucun résultat. Ceci contribue à la dégradation de la qualité des résultats obtenant ainsi une qualité de 0%. En plus, dans le cas où des situations sont décrites par des intentions dont le verbe et/ou la cible sont assez génériques ou spécifiques (*Req5*, *Req6, Req8* et *Req*9 illustrées au Tableau 7), nous obtenons dans certains cas une qualité au dessous des 45%. Ainsi, lorsque le concepteur du système fixe un seuil de paramétrage très élevé dans l'algorithme de prédiction, certains *clusters* ou *états* qui peuvent répondre à l'intention immédiate de l'utilisateur dans son contexte courant ne seront pas sélectionnés, et ceci contribue à la dégradation de la qualité des résultats. Toutefois, dans une situation inédite où, par exemple, il n'existe pas de *cluster* qui est similaire à la requête de l'utilisateur, l'algorithme de prédiction ne retourne aucun résultat. Ceci représente le résultat que nous souhaitons avoir dans une situation pareille présentant ainsi une qualité de résultat très élevé.

Ces résultats indiquent que l'algorithme de *prédiction* a de forte chance de prédire le service le plus approprié à l'intention future de l'utilisateur dans un contexte similaire à son contexte courant. Cependant, il est important lors de la phase de conception de bien définir les ontologies d'une manière très riche et de spécifier le seuil de paramétrage le plus adéquat.

Toutefois, il est à noter que ces résultats sont calculés en fonction de la base de données créée de manière fictive. Ces résultats peuvent ainsi changer en évaluant nos algorithmes avec

de vrais utilisateurs impliqués dans les tests et sur de vraies données tirées d'un réel cas d'étude dans une entreprise. L'évaluation des résultats obtenus démontre ainsi l'intérêt du processus de prédiction de services que nous proposons dans le cadre de ce chapitre. Nous pensons que ce processus proposé permet réellement de prédire le service qui correspond au mieux aux besoins futurs de l'utilisateur.

8.4. Conclusion

Nous proposons dans ce chapitre un processus de prédiction fondé sur une approche intentionnelle et contextuelle, afin de cacher la complexité des SIP. Ce processus permet d'anticiper les besoins futurs de l'utilisateur, afin de lui proposer un service pouvant l'intéresser d'une manière moins intrusive. Nous pensons ainsi contribuer à l'amélioration de la transparence des SIP et à la productivité grâce à une vision centrée sur l'utilisateur.

Ce processus de prédiction de services met en évidence le comportement proactif et d'anticipation de notre vision intentionnelle et contextuelle SIP. Nous croyons fermement qu'une approche de prédiction intentionnelle peut répondre aux exigences de transparence et d'homogénéité, nécessaires à l'acceptation complète du SIP.

Chapitre 9. ARCHITECTURE DE GESTIONNAIRE DE SIP

9.1. INTRODUCTION

Dans la perspective de mettre en place notre vision intentionnelle et contextuelle des SIP (*cf.* Chapitre 4), nous avons proposé dans le Chapitre 5 un cadre conceptuel, que nous avons nommé « *espace de services* » (Kirsch-Pinheiro et al., 2013)(Najar et al., 2013a). Ce cadre permet de masquer l'hétérogénéité des environnements pervasifs et de spécifier les fonctionnalités qui seront proposées et sous quelles conditions, de manière indépendante des technologies. Selon ce cadre, un utilisateur interagit avec le SIP à travers ses espaces de services dans l'objectif d'avoir des services qui répondent à ses besoins dans un contexte donné et en toute transparence. Cependant, pour que notre vision des SIP soit développée, il est primordial de mettre en place une architecture offrant le moyen de gérer le changement de contexte de l'utilisateur et qui offre des mécanismes de découverte et de prédiction de services nécessaires pour supporter et satisfaire l'utilisateur selon son contexte et son intention. Cette architecture, que nous avons appelé *architecture de gestionnaire de SIP*, intègre nos différentes propositions présentées au cours de cette thèse pour la construction d'un SIP transparent et centré utilisateur, à savoir la description de services (*cf.* Chapitre 6), la découverte de services (*cf.* Chapitre 7) et la prédiction de services (*cf.* Chapitre 8).

Dans ce chapitre, nous présentons tout d'abord les éléments nécessaires pour le bon fonctionnement de cette architecture (*cf.* section 9.2). Ensuite, nous détaillons, dans la section 9.3, notre architecture en présentant ses différents modules et répertoires.

9.2. LES PREREQUIS DE L'ARCHITECTURE DE GESTIONNAIRE DE SIP

Pour le bon fonctionnement de notre architecture de gestionnaire de SIP, certains prérequis doivent être établis au préalable. A ce niveau, on peut mentionner que la gestion de contexte et la découverte de services, par exemple, nécessitent un ensemble de modèles et d'ontologies pour pouvoir fonctionner.

En effet, et comme nous l'avons décrit dans le Chapitre 6, pour pouvoir gérer le contexte de l'utilisateur il faut avoir *décrit les différents éléments et sujets de contexte que le concepteur du système souhaite capturer* et avoir *défini le moyen de le faire* (technologie utilisée). Ceci nécessite la description au préalable d'une ontologie *multi-niveaux de contexte* (*cf.* Chapitre 6) qui va décrire tous les concepts de contexte qui sont jugés pertinents dans les espaces de services déterminés. Cette ontologie multi-niveaux de contexte est exploitée, par la suite, par le *gestionnaire de contexte,* qui sera détaillé dans la section 9.3.2*,* afin d'aider à la capture et à la description de contexte courant de l'utilisateur, mais aussi des services et des capteurs.

De plus, cette architecture offre des modules qui interagissent avec un répertoire de services sémantiques, détaillé à la section 9.3.3. Ces modules de découverte et de prédiction de services permettent de sélectionner et de prédire le service qui répond au mieux à l'intention immédiate et future de l'utilisateur dans un contexte donné. En conséquence, pour que ces modules fonctionnent, il est nécessaire que le concepteur du SIP établisse toutes les descriptions sémantiques des services (*cf.* Chapitre 6) spécifiées dans les différents espaces de services de l'utilisateur. En effet, une description sémantique des services en incluant les informations intentionnelles et contextuelles doit être réalisée lors de la conception de l'espace de services afin de permettre l'exploitation de cet espace à travers des mécanismes de découverte et de prédiction de services afin de satisfaire les besoins de l'utilisateur.

9.3. L'ARCHITECTURE DE GESTIONNAIRE DE SIP

Nous présentons dans cette section, l'architecture de gestionnaire de Systèmes d'Information Pervasifs. Comme le montre la Figure 80, notre architecture est composée de cinq modules principaux exploitant la notion d'espace de services, à savoir :

- ***Module de gestion de requête*** **(1)** : responsable du traitement et de l'enrichissement de la requête exprimant le besoin de l'utilisateur (*cf.* section 9.3.1) ;
- ***Module de gestion de contexte*** **(2)** *:* responsable de la gestion des informations contextuelles capturées de l'utilisateur et des entités qui composent l'espace de services également (services et capteurs) (*cf.* section 9.3.2) ;
- ***Module de découverte de services*** **(3)** *:* responsable de la découverte et de la sélection du service qui répond au mieux au besoin immédiat de l'utilisateur dans son contexte courant (*cf.* section 9.3.4) ;
- ***Module d'apprentissage*** **(4)** : responsable de la gestion de l'historique de l'utilisateur afin de déterminer son modèle de comportement (*cf.* section 9.3.5) ;
- ***Module de prédiction de services*** **(5)** : responsable de la prédiction du service le plus approprié qui peux répondre à un besoin futur de l'utilisateur (*cf.* section 9.3.6).

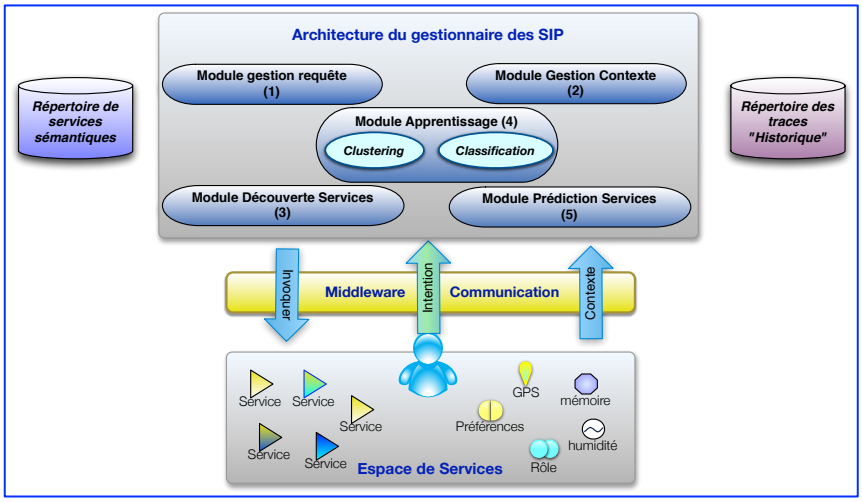


Figure 80. Architecture de gestionnaire de SIP

De plus, notre architecture se base sur un ensemble de répertoires, à savoir :

- ***Répertoire de services sémantiques*** contenant les descriptions de services, les ontologies nécessaires et les descriptions contextuelles (*cf.* section 9.3.3) ;
- **Répertoire de *traces (historique de l'utilisateur)*** contenant l'historique de l'utilisateur ainsi que les *clusters* et le schéma de comportement (*cf.* section 9.3.5).

Les différents modules de l'architecture de gestionnaire de SIP, illustrés à la Figure 80, interagissent entre eux pour recevoir et/ou communiquer des informations, exécuter des opérations, etc. afin d'accomplir leurs responsabilités. Ces différents modules sont par la suite transformés en composants, selon une notation UML. Nous numérotons les modules dans la Figure 80 afin de pouvoir facilement repérer le composant correspondant dans la Figure 81. Ces composants représentent la mise en œuvre des modules de l'architecture, comme l'illustre la Figure 81. Cette figure détaille ainsi la communication entre ces différents composants selon un diagramme de composants UML. Nous détaillons chacun de ses composants, ainsi que les relations entre eux, dans les sections suivantes.

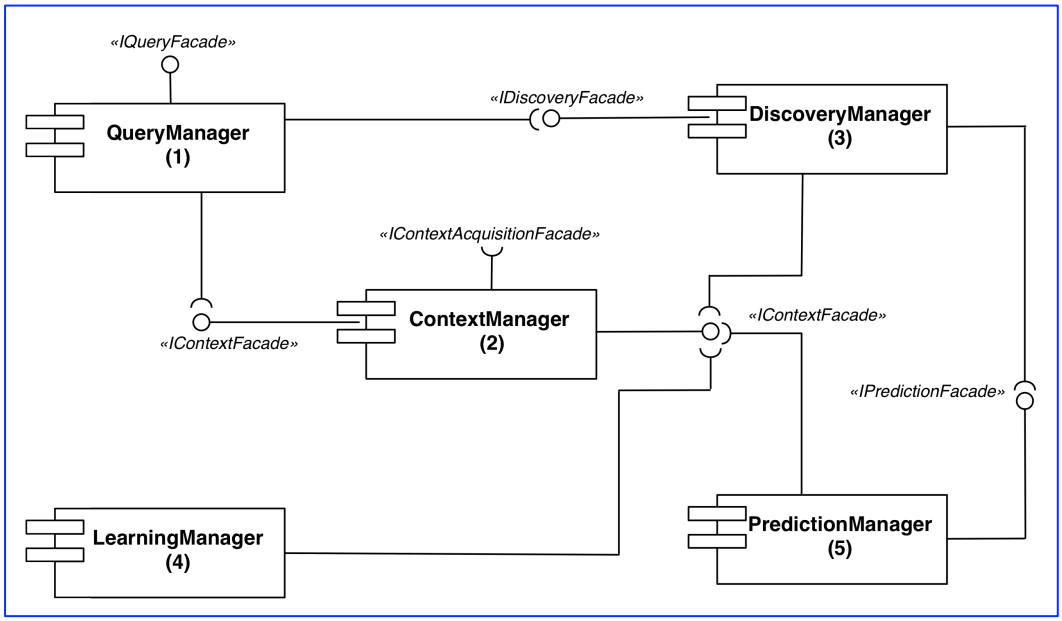


Figure 81. Diagramme de composant de l'architecture de gestionnaire de SIP

Les répertoires de services sémantiques et de traces ne sont pas représentés comme des composants dans notre architecture, mais intégrés au composant qui les manipulent. Ce choix permet de garder plus de souplesse et de flexibilité à l'architecture.

Nous pouvons observer, à partir de cette Figure 81, l'utilisation de plusieurs *façades*. Celles-ci correspondent aux interfaces visibles à l'extérieur des modules de l'architecture. A travers ces interfaces, le module en question arrive à communiquer avec les autres modules de l'architecture et avec des composants externes tels que l'application de capture de contexte, l'interface Web pour la description de la requête de l'utilisateur, etc. L'utilisation de la notion de façade s'inspire du patron de conception façade *(Pattern Facade)*. L'objectif de ce patron

de conception est de *simplifier l'accès aux éléments internes d'un composant à travers une interface visible et reconnue à l'extérieur* (Gamma et al., 1994). Ceci va permettre de réduire les dépendances entre les composants et ainsi améliorer la maintenance.

Dans les sous sections suivantes, nous détaillons chacun de ces différents composants avec ses *façades* ainsi que les différents répertoires exploités et manipulés par notre architecture de gestionnaire de SIP.

9.3.1. Module de gestion de requête (1)

Le module de *gestion de requête* est le module responsable du traitement de la requête de l'utilisateur. En effet, dans le cadre de l'espace de services, l'utilisateur a des besoins qu'il souhaite satisfaire. Pour exprimer son besoin, l'utilisateur formule une *requête* exprimant son intention suivant le modèle de (Prat, 1997). Ainsi, la requête formulée par l'utilisateur à travers une interface Web est par la suite envoyée au module de *gestion de la requête*.

Dans ce cadre, la tâche de l'utilisateur se limite à formuler sa requête sous forme d'intention. Le rôle du module de gestion de la requête est de *récupérer cette requête*, l'*enrichir par le contexte courant* de cet utilisateur, la *décrire sous format XML* afin de *l'envoyer ensuite pour une découverte de services* qui répond au mieux à cette requête.

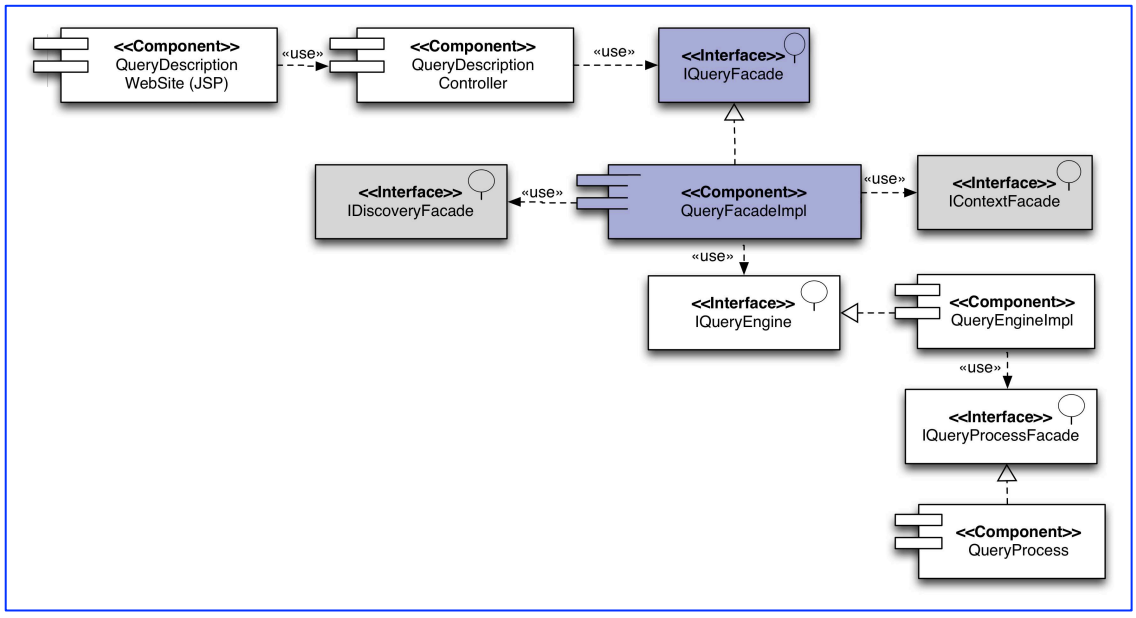


Figure 82. Diagramme de Composant de la description de la requête de l'utilisateur

La Figure 82 présente un aperçu du fonctionnement du *module de gestion de la requête* représenté en UML. Cette modélisation se base sur le paradigme *Model-View-Controller* (MVC) (Reenskaug, 2003). Ceci va permettre de répondre aux besoins des applications interactives tout en séparant les problématiques liées aux différents composants au sein de leur architecture respective, assurant ainsi la clarté de cette architecture. Le composant *QueryDescriptionWebSite* représente la *vue* permettant la formulation de la requête de

l'utilisateur. Cette requête est envoyée par la vue au composant *QueryDescriptionController* représentant le *contrôleur* responsable de la réception de cette requête et la suite de son traitement. Finalement, l'interface *IQueryFacade* représente le *modèle* en charge de traiter concrètement cette requête. C'est à ce niveau que commence le rôle du module de gestion de la requête. Pour accomplir ces fonctions, ce module est lui-même composé par différents sous modules (*cf.* Figure 82), qui se partagent les responsabilités. Premièrement, nous observons la présence d'un composant *QueryFacadeImpl* responsable de la gestion de la requête de l'utilisateur en l'enrichissant par son contexte courant. Ce composant fait appel à deux façades différentes : (i) *IContextFacade* représentant une façade assurant la communication avec le *module de contexte* pour demander et recevoir la description de contexte courant de l'utilisateur (*cf.* section 9.3.2 pour plus de détails) ; et (ii) *IDiscoveryFacade* assurant la communication avec le module de découverte de services pour lui envoyer la description de la requête de l'utilisateur représentant son intention enrichie par son contexte.

En ce qui concerne l'expression de l'intention sur l'interface Web, il est important de souligner ici qu'il existe plusieurs travaux qui s'intéressent au traitement et à la reformulation de cette expression de l'intention, tels que les travaux de (Aljoumaa et al., 2011). Ces auteurs présentent une solution basée sur les ontologies pour analyser les requêtes des utilisateurs écrites en langage naturel. Ce travail permet de reformuler et d'étendre la requête avec de nouveaux concepts en se basant sur des ontologies de verbes et des ontologies de cibles. Cette solution représente ainsi un moyen de guider l'utilisateur à enrichir sa requête et de l'aider à mieux exprimer son besoin. En s'inspirant de ces travaux, une évolution potentielle de notre interface Web est envisageable afin de mieux aider les utilisateurs.

De plus, ce travail de thèse se situe dans le cadre de l'ingénierie des *Systèmes d'Information* (SI). Un SI représente un univers fermé dans un domaine précis n'autorisant pas un comportement ouvert sur des intentions et des cibles non-autorisées ou inconnues auparavant, en raison de leur importance stratégique dans les entreprises. De plus, en dehors d'un tel système, l'expression de l'intention à l'aide d'un ensemble prédéfini de termes est difficilement imaginable, laissant une part trop importante à l'ambigüité. Celle-ci vient, en réalité, des utilisateurs : dans un environnement pervasif ouvert, le profil des utilisateurs n'est pas forcément connu à l'avance et leur mode d'expression peut beaucoup varier. Ainsi, la variabilité dans l'expression des intentions de l'utilisateur dans le domaine des SI reste plus gérable dans ce cadre puisqu'elles sont définies au préalable.

9.3.2. Module de gestion de contexte (2)

Le *module de gestion de contexte* est l'élément de l'architecture responsable de la gestion des informations contextuelles de l'utilisateur et des entités du SIP (services et capteurs). Autrement dit, il se charge de collecter et de modéliser les informations contextuelles courantes des utilisateurs qui interagissent avec le SIP à travers leurs espaces de services (*cf.* Chapitre 5). Il gère également les contextes des services et des capteurs contenus dans le SIP. Le rôle principal de ce module est de cacher la complexité de la gestion de contexte en proposant une manière uniforme d'accéder aux informations contextuelles. A partir des

travaux effectués dans le domaine de la gestion de contexte (Baldauf et al., 2007) (Chaari et al., 2008b) (Paspallis et al., 2008) (Preuveneers et al., 2009), nous pouvons considérer que plusieurs architectures et plateformes de gestion de contexte existent aujourd'hui. A partir de ces architectures, la gestion de contexte se base essentiellement sur : (i) l'acquisition des informations contextuelles à partir des capteurs, (ii) le traitement et la modélisation de ces informations, (iii) leur interprétation et finalement (iv) leur stockage.

En s'inspirant de ces travaux, notre module de *gestion de contexte* est conçu pour recevoir, du composant responsable de l'acquisition de contexte à partir des capteurs (*cf.* Définition 5) physiques et logiques, les informations contextuelles brutes relatives à chaque utilisateur. Ces informations sont par la suite modélisées et interprétées en respectant le modèle de contexte représenté par l'ontologie multi-niveaux de contexte (*cf.* section 6.4.1). Finalement, cette modélisation de contexte est stockée dans un répertoire de contexte.

Ainsi, et afin d'accomplir ses responsabilités, le module de gestion de contexte se divise en plusieurs sous modules, répartis en façades et en sous modules principaux, représentés sous forme de composants comme l'illustre la Figure 83. D'une part, les façades de ce module de gestion de contexte occupent la même fonction que dans le module de gestion de la requête. Elles isolent le contenu des modules et permettent de rendre uniforme l'accès à leurs fonctionnalités. D'autre part, les sous modules principaux (*descripteur de contexte*, *interpréteur de contexte*) se chargent de (i) décrire le contexte de l'utilisateur selon un modèle de contexte bien défini (*cf.* Chapitre 5) et (ii) interpréter ces informations contextuelles en se basant sur un ensemble de règles.

Plus spécifiquement, et en observant la Figure 83, le module de gestion de contexte, représenté par le composant *ContextFacadeImpl*, se déclenche périodiquement pour collecter et représenter le contexte courant de l'utilisateur, comme suit :

- ***Collecte les informations du contexte courant de l'utilisateur* et ceux des capteurs et des services** à partir de la f*açade d'acquisition de contexte* représentée par l'interface *IAcquisitionContextFacade*. Cette façade d'acquisition joue le rôle de médiateur entre les composants responsables de collecter le contexte à partir des capteurs logiques et physiques et entre le composant *ContextFacadeImpl* qui va lancer le traitement de ces donnés.
- ***Les informations contextuelles acquises sont par la suite envoyées par la façade d'acquisition au sous module principal de description de contexte*** représenté par le composant *ContextDataManagerImpl*. Ce sous module se charge de représenter les données brutes reçues sous une forme sémantique plus enrichie en se basant sur un modèle de contexte (*cf.* section 6.4.1). Il se base sur la façade de persistance, représentée par l'interface *IPersistenceManager,* pour charger le modèle de contexte et pour stocker la description de contexte dans le répertoire de contexte.

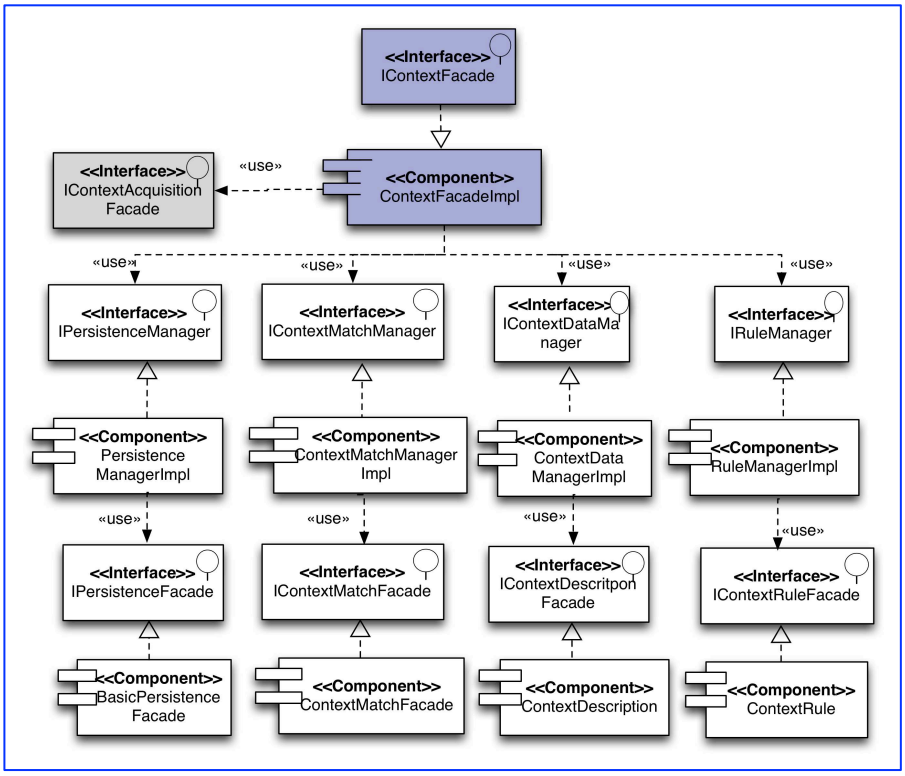


Figure 83. Modèle de Composant du module de gestion de contexte

A part cette tâche, le module de gestion de contexte est sollicité pour :

- ***Déduire de la connaissance à partir des informations contextuelles observées par le capteur***. En effet, ce module contient un sous module responsable de l'interprétation des informations de contexte, le composant *RuleManagerImpl*. Ce composant se charge d'appliquer un ensemble de règles sur les informations contextuelles dans l'objectif d'en déduire de la connaissance.
- ***Communiquer la description de contexte courant de l'utilisateur, du service et du capteur***. Par exemple, le module de gestion de la requête accède au module de gestion de contexte pour solliciter la description de contexte courant de l'utilisateur afin d'enrichir sa requête, à travers la façade *IContextFacade*.
- ***Mettre en correspondance des éléments de contexte***. Le module de gestion de contexte propose un sous module de mise en correspondance entre des éléments de contexte (composant *ContextMatchManagerImpl*). Ce composant est sollicité par le module de découverte de services à travers l'interface *IContextFacade* et sera détaillé dans la section 9.3.4.

9.3.3. Répertoire de services sémantiques

Lors de la phase de conception de l'*espace de services* (*cf.* Chapitre 5), le concepteur se charge de spécifier les services du SIP qu'il juge pertinents dans le domaine cible. Ces services sont décrits sémantiquement par le concepteur, qui est le fournisseur des services, à l'aide du descripteur contextuel et intentionnel défini dans le Chapitre 6.

A partir de l'interface Web « *contextual intentional service description* », le fournisseur des services peut charger la description sémantique du service écrite en OWL-S. Cette description ne comprend ni les informations intentionnelles du service, ni ses informations contextuelles. Ensuite, à partir de cette interface, il peut : (i) introduire les intentions que ce service permet de satisfaire et (ii) indiquer l'URL de la description de contexte de ce service. En sauvegardant ce travail, une description intentionnelle et contextuelle du service est établie en OWL-SIC (*cf.* Chapitre 6). Cette description de services est par la suite enregistrée dans un *répertoire de services sémantiques*. Ce répertoire représente une base de stockage des différents services intentionnels et contextuels. Il est structuré comme suit :

- *Répertoire de services* contenant l'ensemble des descriptions de services intentionnelles et contextuelles décrits en OWL-S.
- *Répertoire d'ontologies* contenant toutes les ontologies nécessaires pour la description, la découverte et la prédiction de services. Il contient les ontologies des cibles, des verbes (*cf.* section 6.3.1.3) et l'ontologie de contexte (*cf.* section 6.4.1).
- *Répertoire de contexte* contenant toutes les descriptions de contexte courant de l'utilisateur, des services et des capteurs.

Le répertoire de services sémantiques joue le rôle, tel que l'annuaire de service dans l'architecture SOA (Papazoglou et Heuvel, 2007), d'intermédiaire entre le fournisseur de service et le client demandeur d'un service. D'une part, et comme nous l'avons mentionné ci-dessus, le fournisseur de service interagit avec ce répertoire pour y stocker les fichiers de description de services sémantiques. D'autre part, l'utilisateur interagit avec ce répertoire pour y chercher le service qui satisfait au mieux ses intentions dans un contexte donné. En effet, lorsqu'une requête de l'utilisateur, une fois traitée par le module de gestion des requêtes, arrive au module de découvertes de services (*cf.* section 9.3.4), celui-ci lancer la procédure de sélection du service le plus approprié à partir de l'ensemble des descriptions de services enregistrées dans ce répertoire.

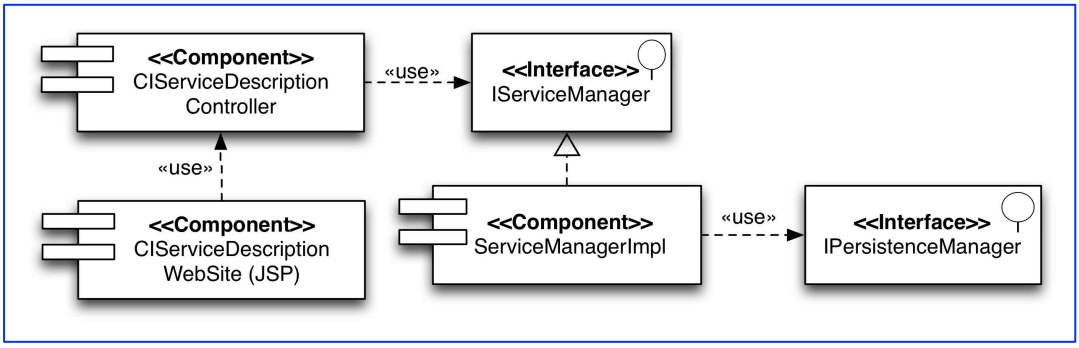


Figure 84. Diagramme de Composant de la description de service

La Figure 84 présente un aperçu du traitement des descriptions des services et des ontologies répertoriées dans le répertoire de services sémantiques représenté en UML. Cette modélisation se base sur le paradigme MVC comme nous l'avons présenté dans la section

9.3.1. Ce *module de traitement des descriptions de services et des ontologies,* comme l'illustre la Figure 84, se compose de :

- Un composant *CIServiceDescription Website* : représente le site Web permettant au fournisseur de service (concepteur des SIP) de manipuler les descriptions de services et les ontologies nécessaires ;
- Un composant *CIServiceDescription Controller :* représente le contrôleur du site Web de description de services et d'ontologies ;
- Une interface *IServiceManager* : représente le point d'accès à ce module ; Elle offre des méthodes pour manipuler les descriptions de services et les ontologies gérées par le composant *ServiceManager*Impl.

L'interface *IPersistenceManag* assure le maintien des descriptions de services et des ontologies en offrant des méthodes d'*écriture*, de *lecture*, de *rajout*, de *suppression*, de *chargement*, etc. Elle va être utilisée par la majorité des modules que nous présentons dans ce chapitre. Toutefois, nous ne présentons pas à ce niveau le composant responsable de la description des ontologies puisqu'il existe plusieurs outils d'édition d'ontologies OWL, à l'instar de Protégé[5]

9.3.4. Module de découverte de services

L'architecture de gestionnaire de SIP intègre un module de *découverte de services* (Najar et al., 2012a) (Najar et al., 2012b). Ce module est basé sur le mécanisme de découverte de services guidé par l'intention et le contexte de l'utilisateur, proposé dans le Chapitre 7.

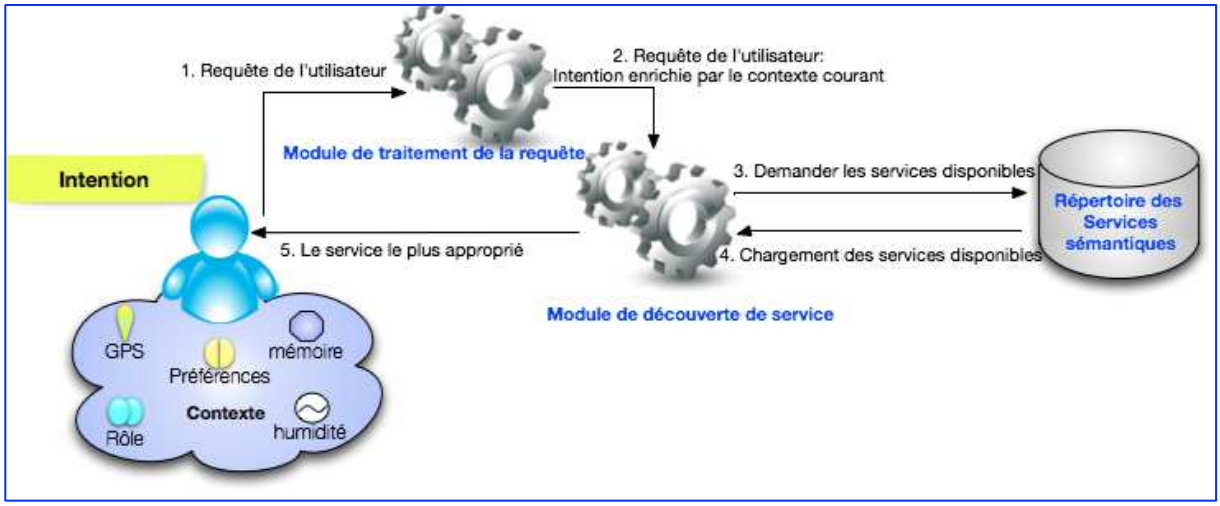


Figure 85. La communication avec le module de découverte de service

Le *module de découverte de services*, se déclenche lorsqu'il reçoit la requête enrichie de l'utilisateur de la part du *module de gestion de la* requête (Figure 85) à travers la façade

[5] *http://protege.stanford.edu/*

représentée par le l'interface *IDiscoveryFacade*. Dès lors, il lance un *algorithme de découverte sémantique des services* (*cf.* Chapitre 7). Cet algorithme effectue un processus de mise en correspondance sémantique, à travers le composant *SearchEngineImpl* illustré à la Figure 86, afin de sélectionner le service le plus approprié à l'utilisateur. Le but de cet algorithme est de classer les services disponibles en fonction de leurs informations contextuelles et intentionnelles. Ensuite, il sélectionne le service jugé le plus approprié par rapport à l'utilisateur. La disponibilité (état) d'un service est déterminée à partir de son contexte courant (Cx_{svi}) fourni par le gestionnaire de contexte. Ce gestionnaire de contexte se charge de gérer cette dynamique des services (*cf.* section 5.4.3). Celui-ci doit ainsi gérer dynamiquement les états des services disponibles pour le processus de découverte. Différentes recherches ont été menées sur ce sujet proposant différentes stratégies. Cette partie reste un point ouvert dans notre architecture, mais qui ne sera pas traité dans le cadre de cette thèse.

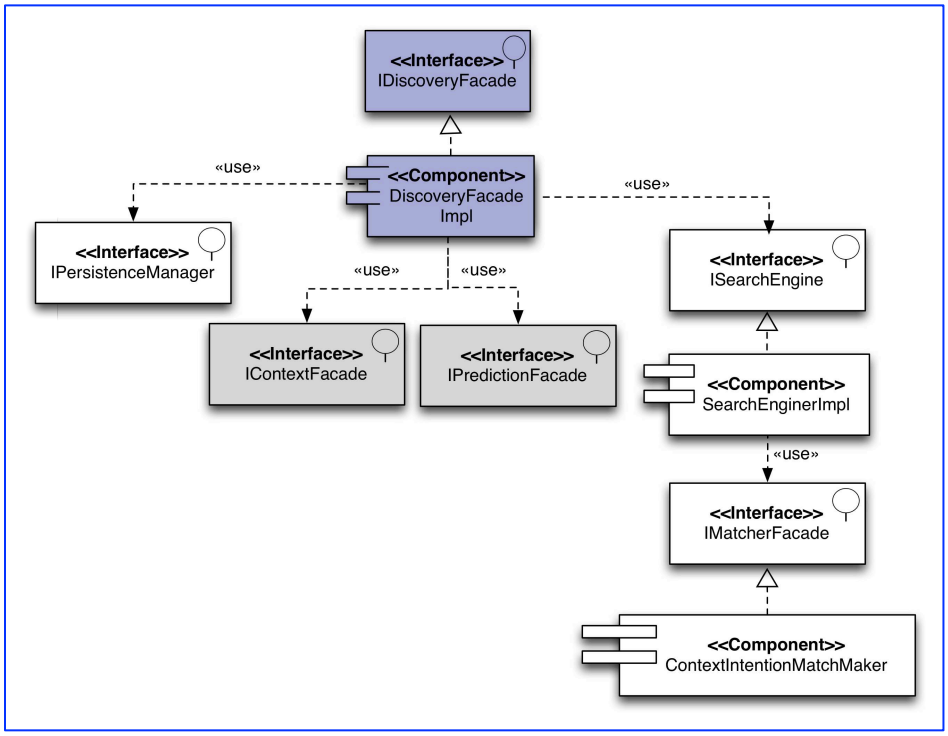


Figure 86. Diagramme de Composant de la découverte de services

La Figure 86 présente un modèle de composant du module de découverte de services. Le point d'accès à ce module représente l'interface *IDiscoveryFacade* implémentée par le composant *DiscoveryFacadeImpl* (*cf.* section 7.3.2.2). L'implémentation de cette interface fait appel à l'interface *ISearchEngine* et aux interfaces représentant les façades *IPersistenceManager, IPredictionFacade* et *IContextFacade*. L'interface *ISearchEngine* est utilisée pour découvrir et sélectionner le service le plus approprié, et d'autre part les façades permettant communiquer avec les autres modules. Le *IPersistenceManager* est responsable de la gestion (lecture, écriture, chargement, suppression, etc.) des descriptions de services et des ontologies. L'interface *IPredictionFacade* représente l'intermédiaire entre le module de découverte de services et le module de prédiction de services, dont le but est de lancer le mécanisme de prédiction de services après une découverte de services. Finalement, la façade

IContextFacade est celle responsable de communiquer le module de découverte de services avec le module de gestion de contexte afin de lancer la procédure de mise en correspondance entre les éléments de contexte.

Cet *algorithme de découverte sémantique des services* (*cf.* section 7.2.2) compare sémantiquement l'intention de l'utilisateur avec les intentions que le service permet de satisfaire. Il communique avec le module de gestion de contexte à travers sa façade *IContextFacade*, pour lancer la mise en correspondance entre le contexte courant de l'utilisateur avec le contexte requis du service. Ensuite, il sélectionne le service le plus approprié répondant à l'intention immédiate de l'utilisateur dans son contexte actuel. Cet algorithme charge à travers la façade de persistance (*IPersistenceManager*) toutes les descriptions sémantiques de services disponibles qui sont répertoriées dans le répertoire de services (*cf.* section 9.3.3).

Il convient de souligner que même si la mise en correspondance intentionnelle s'effectue par le composant *SearchEngineImpl*, la mise en correspondance contextuelle est effectuée par le sous module de matching de contexte (*IContextMatchManager*) du module de gestion de contexte, afin de réduire les risques que l'évolution des éléments à l'intérieur du *module de gestion de contexte* puisse affecter d'avantage les éléments du *module de découverte de services* qui l'utilisent, réduisant ainsi les dépendances entre eux.

Par ailleurs, les implémentations des interfaces illustrées dans la Figure 86, ainsi que les autres, utilisent le patron de conception « *strategy* » pour fournir un changement souple de stratégie dans la perspective d'assurer un caractère flexible, réutilisable et échangeable de notre architecture. Celle-ci doit pouvoir être spécifiée en temps d'exécution. Ainsi, au cours de la phase de démarrage, la stratégie définie au préalable, par un fichier de configuration, est chargée comme étant la stratégie par défaut.

9.3.5. Module d'apprentissage

Après chaque phase de découverte de services, la situation de l'utilisateur est horodatée et stockée dans une base de données afin de générer des traces de l'utilisateur (son historique). En analysant ces situations représentées par le triplet <intention, contexte, service>, d'une façon périodique, le *module d'apprentissage* peut d'inférer les comportements types de l'utilisateur, dans un environnement dynamique (*cf.* section 8.2.2).

Ce module d'apprentissage se charge de déterminer dynamiquement le modèle de comportement de l'utilisateur (*classification*) à partir des *clusters* représentant ses situations similaires (*clustering*) et de l'historique. Il s'agit d'une tâche en arrière plan qui doit être réalisée de manière récurrente (*cf.* section 8.2.2).

La Figure 87 illustre les différents composants et interfaces qui interviennent lors d'une phase d'apprentissage du modèle de comportement de l'utilisateur. Le point d'entrée à ce module est l'interface *ILearningFacade* qui se charge de lancer périodiquement le processus

de *clustering* et de *classification*. L'implémentation de cette interface (*LearningFacadeImpl*) fait appel aux quatre interfaces suivantes :

- *IClassificationEngine* : lance l'algorithme de classification (*cf.* section 8.3.1.2) ;
- *IClusteringEngine* : lance l'algorithme de *clustering* (*cf.* section 8.3.1.1) ;
- *IContextFacade* : la façade responsable de communiquer avec le module de gestion de contexte afin de lancer la procédure de mise en correspondance entre les éléments de contexte nécessaires durant la phase de *clustering* ;
- *IPersistenceManager* : la façade responsable de la gestion des descriptions de services et des ontologies.

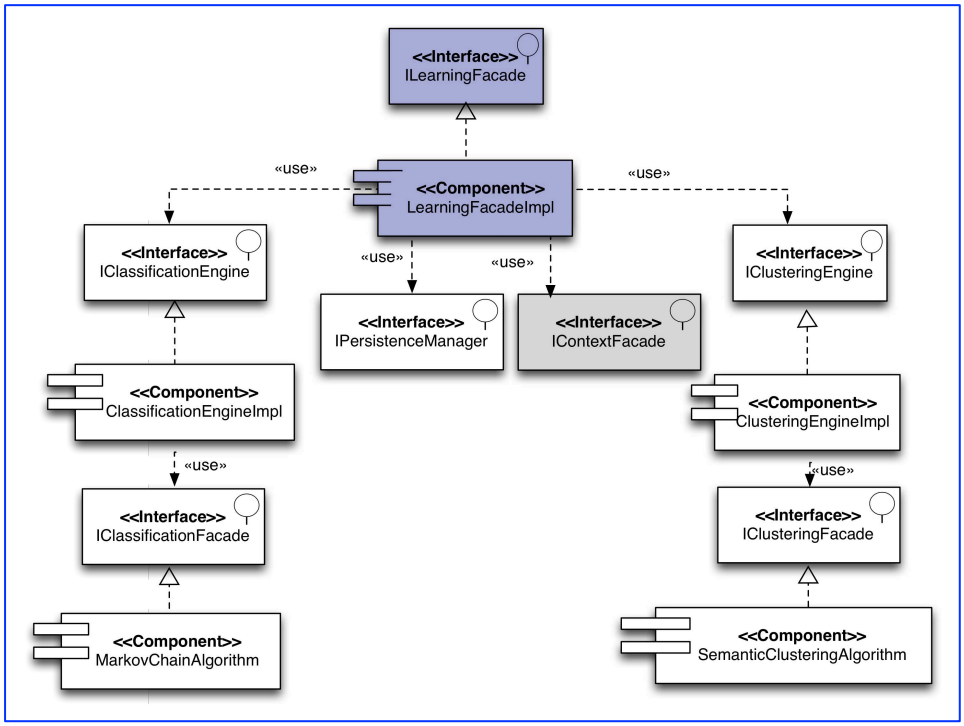


Figure 87. Diagramme de composant de l'apprentissage

9.3.6. Module de prédiction de services

Le *module de prédiction* met en place, dans l'architecture de gestionnaire de SIP, le mécanisme de prédiction proposé dans le Chapitre 8. Ce module tente ainsi de prévoir l'intention future de l'utilisateur afin de lui proposer le prochain service qui peut répondre à cette intention future.

Ce processus se déclenche lorsqu'une phase de découverte de services se déroule, comme l'illustre la Figure 88. Basée sur l'intention de l'utilisateur (I_U) et sur son contexte courant (Cx_U), notre architecture est capable non seulement de sélectionner le service qui répond au mieux au *besoin immédiat* de l'utilisateur (phase de découverte de services), mais également de lui proposer le service suivante le plus probable (phase de prédiction de services) répondant à son *besoin futur*.

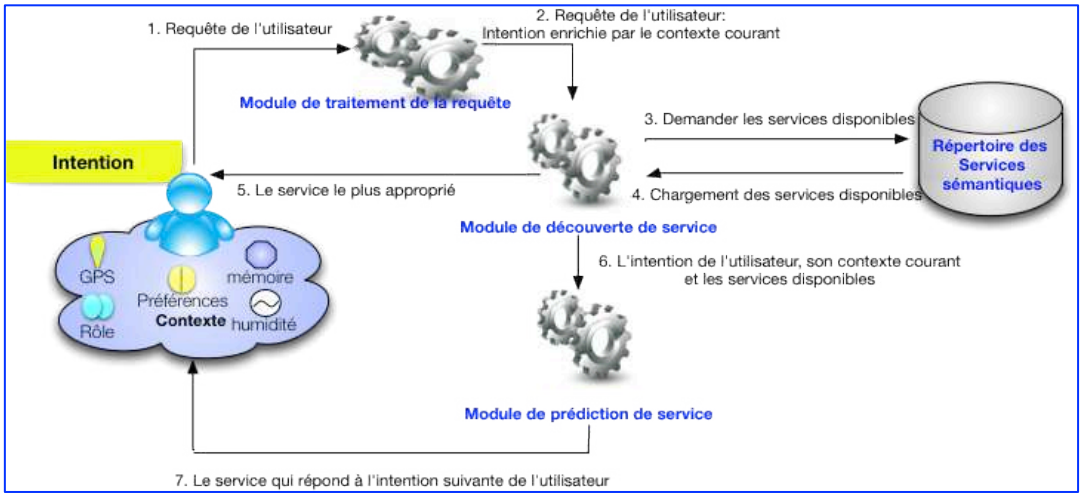


Figure 88. La communication avec le module de prédiction de services

La Figure 89 présente un modèle de composant du module de prédiction de services. Le point d'accès à ce module représente l'interface *IPredictionFacade* implémentée par le composant *PredictionFacadeImpl* qui offre des méthodes de prédiction de services. L'implémentation de cette interface fait appel à l'interface *IPredictionEngine* et aux interfaces : *IPersistenceManager et IContextFacade*.

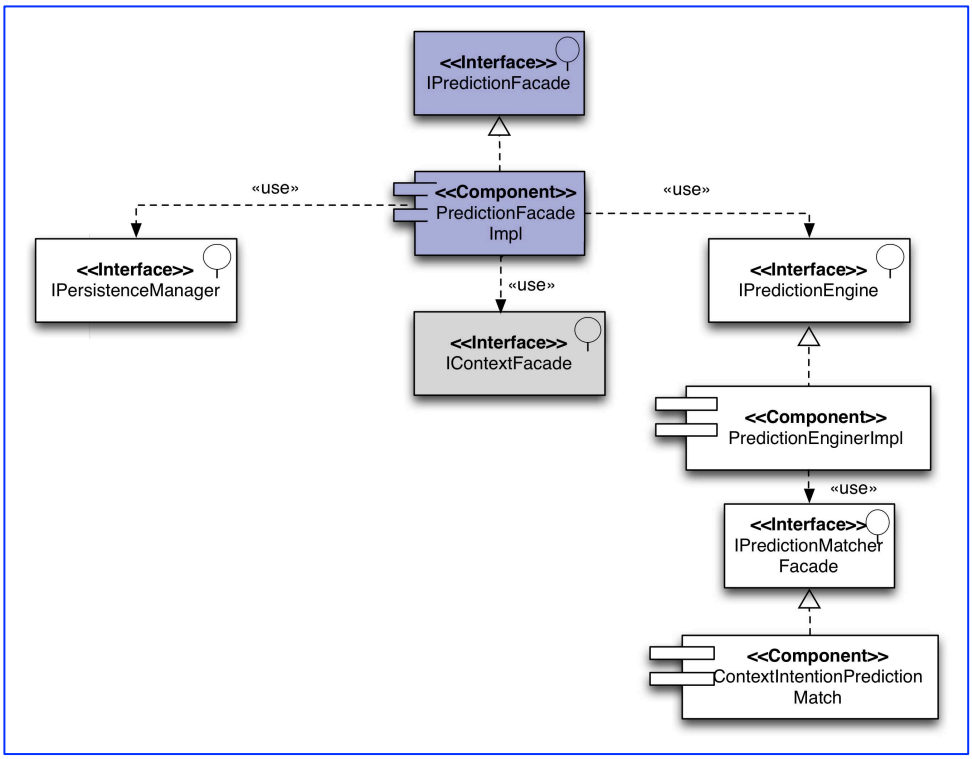


Figure 89. Diagramme de composant de la prédiction de services

L'interface *IPredictionEngine* (*cf.* section 8.3.1.3) est utilisée pour prédire et sélectionner le service suivant le plus approprié. L'interface *IPersistenceManager* permet l'accès aux descriptions de services et aux ontologies. Finalement, l'interface *IContextFacade* est la

façade utilisée pour lancer la procédure de mise en correspondance entre les éléments de contexte auprès du module de gestion de contexte.

Ce module de prédiction de services, détaillé dans le Chapitre 8, se base sur *un algorithme de prédiction de l'intention et du service futur de l'utilisateur* (Najar et al., 2013b) (Najar et al., 2012c) afin de prédire le service suivant le plus approprié.

9.4. Conclusion

Dans ce septième chapitre, nous avons présenté le point de vue système de notre vision intentionnelle et contextuelle des SIP en présentant notre architecture conceptuelle du gestionnaire de SIP. Cette architecture représente la mise en place de notre vision des SIP offrant ainsi des mécanismes permettant de gérer le changement de contexte de l'utilisateur, et des mécanismes de découverte et de prédiction de services nécessaires pour supporter et satisfaire l'utilisateur selon son contexte et son intention.

Chapitre 10. DEMARCHE METHODOLOGIQUE DE CONCEPTION D'UN SIP

10.1. INTRODUCTION

Afin de faciliter et d'organiser la conception de l'espace de services présenté dans le Chapitre 5, nous proposons dans ce chapitre, une démarche de conception supportant le passage de cet espace de services à l'architecture de gestionnaire de services, à destination des concepteurs de SIP et spécialement à la Direction du Système d'Information (DSI). Cette démarche méthodologique sert à les aider à spécifier les fonctionnalités attendues de leur système, ainsi que les informations qui seront capturées par celui-ci pour une meilleure adaptation. Il s'agit d'un processus de conception d'un espace de services et de ses différents éléments dans la perspective de garder le contrôle sur la définition du système et de ses services, tout en permettant la prise en compte d'un environnement hautement dynamique.

Pour effectuer cette démarche, nous nous basons sur une approche *hybride* qui fusionne des méthodes de conception de type descendante (*top-down)* et ascendante (*bottom-up)*. Dans l'approche *descendante*, les besoins de l'utilisateur et la définition des principaux espaces de services sont dérivés à partir d'une analyse et une étude des utilisateurs dans leurs différents espaces de travail possibles. Dans l'approche *ascendante*, les services proposés et les éléments de contexte identifiés sont le résultat d'une synthèse faite sur les différentes fonctionnalités possibles dans le cadre du SI de l'entreprise et sur une analyse des différents capteurs de contexte possibles.

Dans ce chapitre, nous présenterons un aperçu du processus de cette démarche de conception de l'espace de services et nous détaillerons chacune de ses étapes.

10.2. PRESENTATION DE LA DEMARCHE METHODOLOGIQUE

Dans la perspective de représenter un cadre conceptuel pour la conception d'un SIP, nous avons proposé la notion d'*espace de services*. Cette notion permet la définition d'un SIP par la spécification d'un ensemble d'espaces, dans lesquels cohabitent les services offerts par le système et les capteurs qui l'alimentent avec des informations récoltées à partir de l'environnement. Afin de mieux organiser la conception d'un tel espace et de faciliter la tâche du concepteur des SIP, nous proposons une démarche méthodologique de conception de ce cadre conceptuel des SIP afin d'aider et de guider le concepteur dans sa conception des différents espaces de service et de ses différents éléments qu'il juge remarquables et pertinents dans le domaine cible.

D'une manière générale, les démarches méthodologiques de conception sont basées soit sur une approche ascendante (*top-down)*, soit sur une approche descendante (*bottom-up)*. Une

approche *top-down* correspond à une démarche de conception, allant de la spécification au développement. Appliquée à l'orientation service, elle commence par la spécification des processus métiers pour descendre ensuite à la définition des services nécessaires à la réalisation de ces processus. Le point fort de cette approche est qu'elle se focalise essentiellement sur l'objectif principal du système. C'est une approche orientée utilisateur qui exprime clairement les besoins des utilisateurs, sans pour autant se soucier du fonctionnement interne du système en termes de tâches du processus. Néanmoins, la prise de décision en se basant uniquement sur les objectifs et pas en fonction des descriptions réalisées ultérieurement représente l'un des inconvénients majeurs de cette approche. En effet, certains changements peuvent apparaître au cours de la réalisation, ce que n'anticipe pas l'approche descendante qui se base sur une vision globale de départ.

A l'inverse, une approche *bottom-up* suit une démarche de conception ascendante, partant de l'existant. Elle commence par l'analyse de l'existant, afin de déterminer les fonctionnalités existantes du SI. Ces fonctionnalités sont ensuite décrites sous forme de service. Le point fort de cette approche est qu'elle permet l'identification des données mises à disposition par le système afin déterminer les informations qui seront réellement accessibles aux utilisateurs. Toutefois, cette approche se base plus sur le fonctionnel et non sur le métier risque de s'éloigner de l'objectif du système et de ne pas répondre aux besoins de manière satisfaisante.

Les points forts et les faiblesses des approches *bottom-up* et *top-down* nous ont motivé au développement d'une *approche hybride* exploitant les avantages de chacune de ces approches. La démarche méthodologique que nous proposons repose sur une combinaison entre les approches ascendante et descendante, afin de prendre en compte à la fois sur le métier et le fonctionnel pour permettre la définition de l'espace de services et de ses différents éléments métiers et techniques. La conception d'un SIP à l'aide de la notion d'espace de services commence par la définition des multiples espaces dans lesquels les utilisateurs vont évoluer. Chaque espace est défini en fonction des entités actives (*cf.* section 5.4.1), représentant les services offerts aux utilisateurs et des entités passives, permettant l'observation de l'environnement. Les entités actives (services) sont ainsi définies en fonction de l'intention qu'elles doivent satisfaire et du contexte dans lequel ces intentions émergent. Les entités passives (capteurs) (*cf.* section 5.4.2) sont importantes puisqu'elles sont responsables de la capture des informations contextuelles dont le système aura besoin pour l'adaptation. Leur définition délimite la notion de contexte, spécifiant les informations considérées comme pertinentes. L'objectif est de permettre la prise en compte de ce contexte afin de proposer aux utilisateurs des services qui leur correspondent au mieux.

La démarche proposée, illustrée à la Figure 90, s'organise ainsi en multiples étapes, décrites ci-dessous. Ces étapes sont décrites dans l'ordre qui nous semble le plus pertinent du point de vue du concepteur d'un SIP. Nous commençons par délimiter les espaces de services et ensuite par l'identification des fonctionnalités que doit fournir le système, qui se traduisent par la spécification des services offerts dans ces espaces.

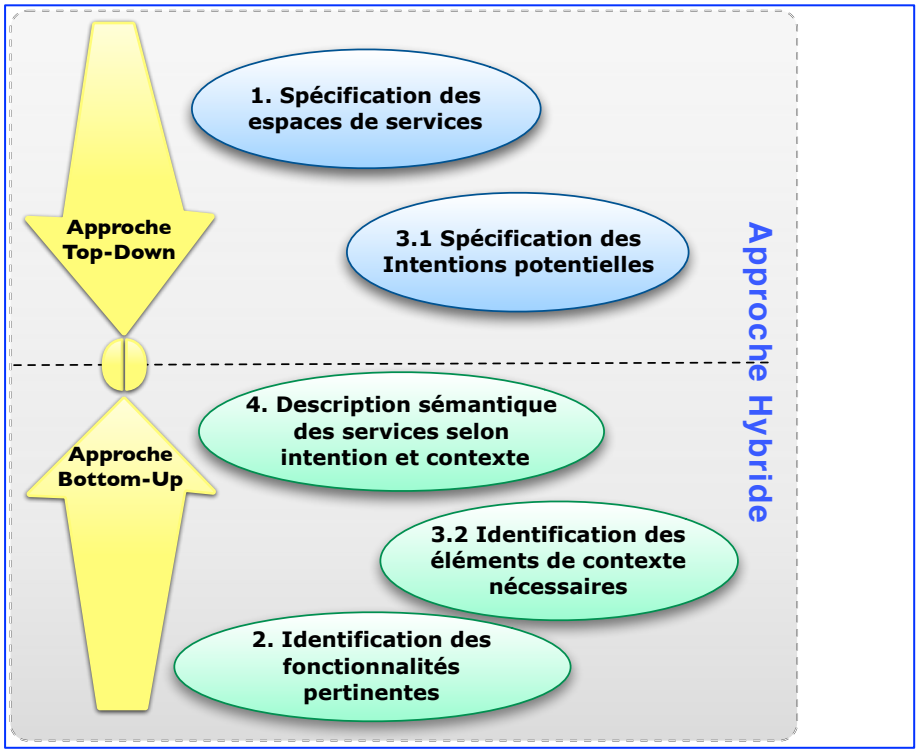


Figure 90. Vue schématique de la démarche méthodologique de conception d'un SIP : Approche hybride

1. Spécification des espaces de services :

Cette étape vise l'analyse et la spécification des principaux espaces de services par l'observation de l'utilisateur. Elle se base sur une approche *descendante* qui part du niveau métier et du domaine d'application du système. Le résultat est une liste des différents espaces de services les plus remarquables et souhaités pour le SIP.

2. Identification des fonctionnalités pertinentes :

Cette étape permet d'identifier les différentes fonctionnalités considérées comme pertinentes pour le SIP. Cette étape suit ainsi une approche *ascendante* qui part de l'existant au niveau fonctionnel. Les fonctionnalités existantes sont d'abord identifiées, et puis classées en fonctionnalités d'infrastructure, métiers et utilitaires. La correspondance entre les fonctionnalités identifiées et les services techniques est ensuite effectuée.

3. Identification du couple <Intention, Contexte> :

Cette étape permet d'identifier, d'une part, les intentions potentielles de l'utilisateur et qui sont réalisables par le système (sous-étape 3.1), ainsi que les éléments de contexte nécessaires qui jouent un rôle central dans le processus d'adaptation du système et qui peuvent influencer la manière de satisfaire les intentions (sous-étape 3.2). A cette étape, le concepteur peut commencer par la sous-étape qu'il souhaite. Il peut commencer par identifier les intention et par la suite les éléments de contexte qui sont susceptibles d'influencer chacune de ces intentions, ou bien il peut commencer par identifier les éléments de contexte qu'il est capable de détecter, et ensuite détermine les intentions qui sont réalisables.

3.1 Spécification des intentions potentielles :

Cette étape se base sur une analyse plus approfondie des besoins potentiels de l'utilisateur et leur spécification sous forme d'intention. Cette étape suit ainsi une approche *descendante* qui part de l'utilisateur et de ses besoins, afin de mieux comprendre le fonctionnement que doit avoir le système du point de vue utilisateur. Le résultat est une liste des intentions potentielles de l'utilisateur dans ses différents espaces de service.

3.2 Identifications des éléments de contextes nécessaires :

Cette étape vise à identifier les différents éléments de contexte qui sont pertinents par rapport aux services pouvant être proposés dans l'espace de services. Cette étape se base sur une approche *ascendante* qui part des technologies disponibles permettant d'observer et de capturer les éléments de contexte nécessaires.

4. **Description sémantique des services selon l'intention et le contexte :**

Cette étape concerne l'incorporation de la technologie du Web sémantique pour décrire les fonctionnalités existantes (2) sous forme de service sémantique. Cette description est enrichie, d'une part, par l'intention (3.1) auquel le service peut répondre, et d'autre part, par le contexte (3.2) dans lequel ce service est valide et exécutable.

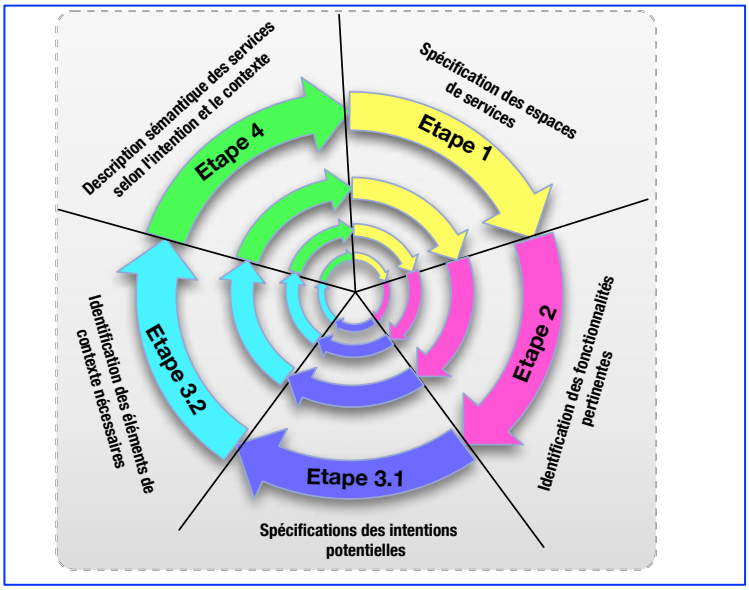


Figure 91. Démarche méthodologique de conception d'un SIP : processus incrémental et évolutif

Cette démarche méthodologique de travail ne suit pas un processus linéaire et figé. Au contraire, elle représente un processus de création incrémental et évolutif. Comme l'illustre la Figure 91, cette démarche est représentée sous forme de spirale, ce qui va permettre au concepteur de faire des aller-retour sur chacune des étapes de la démarche de la conception pour compléter et/ou modifier les résultats trouvés.

La section suivante décrit en détails la démarche proposée en spécifiant les différentes étapes de ce processus de conception du SIP.

10.3. LES ETAPES DU PROCESSUS DE CONCEPTION D'UN SIP

Après avoir énoncé, d'une manière générale, les différentes étapes de notre démarche méthodologique de conception d'un SIP à travers un espace de services, nous détaillons dans cette section le déroulement de chacune de ces étapes.

10.3.1. Etape 1 : Spécification des espaces de services

L'étape de spécification des différents espaces de services réside donc dans l'analyse et la spécification des espaces de services. Cette étape permet d'établir au préalable, et d'une façon générale, les différents espaces de services les plus importants du point de vue d'un *utilisateur mobile*. Il s'agit de déterminer conceptuellement les espaces de services au sein desquels l'utilisateur devra interagir avec le SIP. Il s'agit d'une approche de raisonnement par observation d'une ou plusieurs catégories d'utilisateur (définie par leur rôle) que le concepteur a ciblé par rapport à leur mobilité. Ainsi, il est capable de définir les espaces de services où le système peut fournir quelque chose. L'espace de services est représenté comme une *boite noire*. Ainsi, il faut déterminer ce qu'on cible comme population et leur mobilité afin de concentrer nos efforts là-dessus.

Dans le cadre d'un SIP, le concepteur est amené d'abord à déterminer son domaine cible. Par la suite, il doit considérer les espaces de services les plus pertinents, dans ce domaine, afin que la DSI garde le contrôle malgré le dynamisme et l'hétérogénéité de l'environnement pervasif. En effet, le concepteur doit se mettre dans la peau de l'utilisateur, observer son comportement et ses besoins en prenant en considération sa mobilité, et déterminer ainsi ses espaces de travail les plus importants. Ces différents espaces se définissent aussi par rapport à son métier. Chaque espace est identifié, en remarquant des changements dans les besoins de l'utilisateur, dans la réalisation de ses besoins, dans la disponibilité ou la non disponibilité de certains éléments des services.

Chaque espace de services identifié sera par la suite labellisé. Le label attribué doit être significatif et représentatif de l'espace dans lequel se trouve l'utilisateur, par exemple : « Entreprise », « Maison », « Chez le client », « A l'extérieur », etc.

Le résultat de cette étape est une liste des différents espaces de services qui vont être détaillés par la suite en termes d'intentions potentielles de l'utilisateur (ses besoins), d'éléments de contexte observables et des services offerts (les réalisations des intentions).

10.3.2. Etape 2 : Identification des fonctionnalités pertinentes

L'étape d'identification des fonctionnalités pertinentes part de l'analyse de l'existant, afin de déterminer les fonctionnalités existantes du SI. Nous partons en réalité du principe qu'un SI traditionnel est déjà mis en place et que le futur SIP ne sera pas construit à partir de rien. Il s'agit surtout d'une évolution (par transformation) d'un SI vers un SIP.

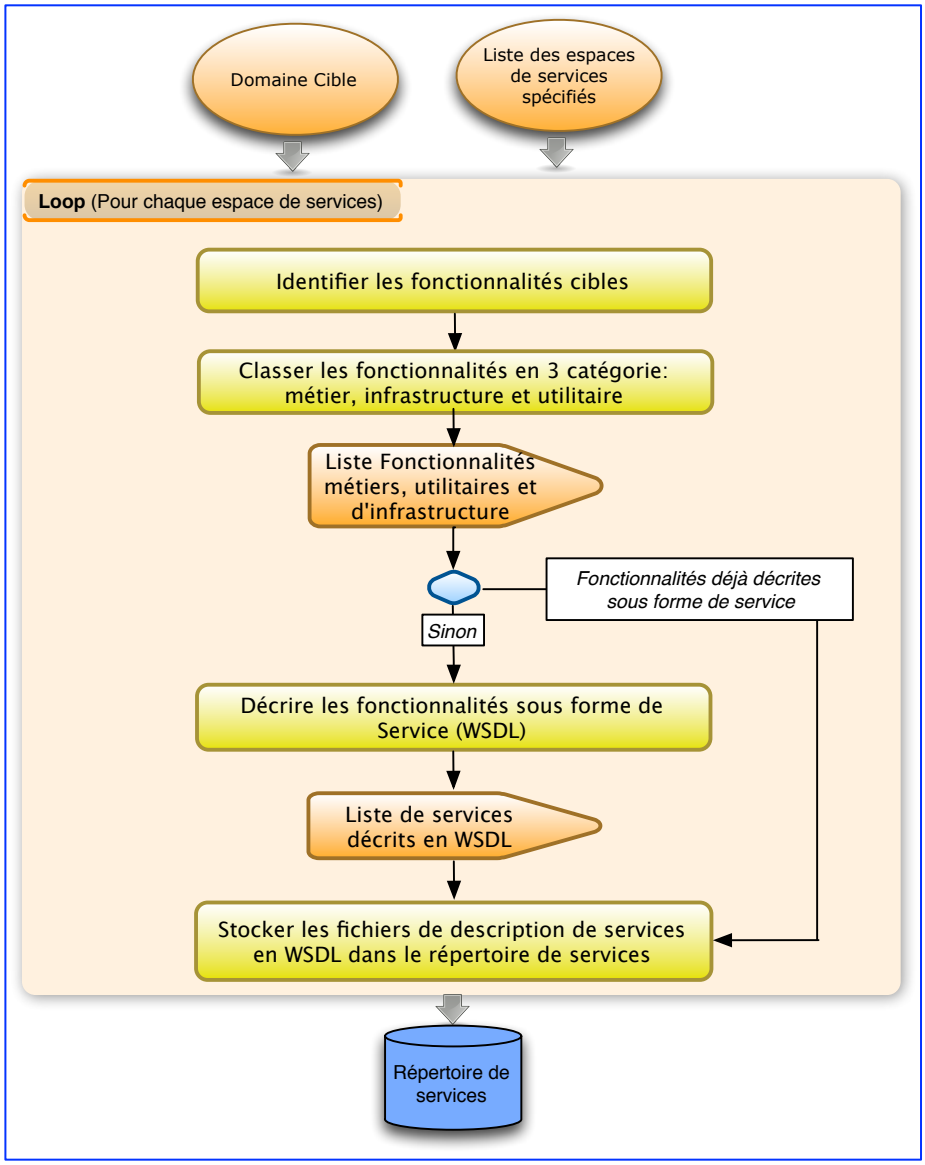


Figure 92. Processus d'identification des fonctionnalités du système décrites en services techniques

Comme l'illustre la Figure 92, cette étape 2 reçoit comme entrée le domaine cible et les multiples espaces de services spécifiés lors de l'étape 1. Pour chaque espace de services, le concepteur commence ainsi par identifier l'ensemble des fonctionnalités qu'il souhaite exposer en tant que partie pertinente d'un SIP. Ces fonctionnalités seront par la suite classées en trois catégories : *métier* (relative au domaine cible, *e.g.* rédaction rapport, organiser réunion, etc.), *infrastructure* (*e.g.* connexion réseau, réglage sécurité, cryptage des données, etc.) et *utilitaire* (*e.g.* faxer un document, imprimer un document, etc.). Cette classification apporte plus de clarté à la nature et au type de la fonctionnalité. Ces fonctionnalités sont par la suite décrites sous forme de services techniques. Ces services techniques définissent les fonctions techniques du système et comment elles se déroulent.

Nous considérons que ces fonctionnalités sont décrites au préalable sous forme de services Web en WSDL. La description WSDL (Christensen et al., 2001) représente une interface

standard indépendante de la plateforme pour la description de service. Néanmoins, dans le cas où certaines fonctionnalités, jugées pertinentes, ne seraient pas décrites sous forme de services, une telle description doit être fournie. Pour ce faire, le concepteur peut se baser sur les *framework*s existants, tels que *Apache Axis SOAP implementation*[6], *Fuse services framework*[7] et *Sun Web service development pack*[8]. Ces *frameworks* se chargent de générer automatiquement le fichier WSDL correspondant à chaque fonctionnalité.

Le résultat de cette étape est donc un ensemble de services techniques décrits en WSDL. Ces services sont par la suite stockés dans le répertoire de services.

10.3.3. Etape 3 : Identification du couple intention et contexte

Dans le cadre de cette troisième étape, nous soulevons deux alternatives de travail. La première alternative part du bas niveau, des fonctionnalités, afin de déterminer les intentions auxquelles ces fonctionnalités peuvent répondre. Cette alternative est le plus souvent choisie par les gens techniques. La deuxième alternative part du haut niveau afin d'éliciter les intentions potentielles de l'utilisateur et les éléments de contexte que le concepteur est capable de trouver vis-à-vis de l'utilisateur. Cette alternative est le plus souvent choisie par les gens métiers. Ainsi, le concepteur peut identifier le couple <Intention, Contexte> à partir de l'*utilisateur* ou à partir des *fonctionnalités*. Cependant, si par exemple le concepteur a choisi l'alternative d'identifier le couple <Intention, Contexte> à partir des fonctionnalités, il est amené ensuite à confronter le résultat qu'il a trouvé avec les intentions et contextes qu'il peut déterminer côté utilisateur, et vis-versa. Ceci représente une étape de validation qui permet de raffiner le résultat trouvé.

Dans le cadre de ce travail, nous avons choisi la première alternative de travail qui élicite le couple <Intention, Contexte> du côté utilisateur. Nous commençons ainsi par l'identification des intentions, étant donné que notre objectif est avant tout de proposer une approche centrée utilisateur. Cet ordre peut être modifié d'un concepteur à un autre.

10.3.3.1. Etape 3.1 : Spécification des intentions potentielles de l'utilisateur

L'étape de spécification des intentions possibles de l'utilisateur vise à analyser et identifier les besoins potentiels des utilisateurs. Cette étape orientée utilisateur permet de lister, pour chaque espace de services identifié précédemment et dans le domaine cible, les différentes intentions qu'un utilisateur peut avoir et qui sont susceptibles d'être réalisables par un SIP. L'identification de ces intentions guidera, par la suite, la DSI dans l'identification des services identifiés capables de répondre à ces intentions.

---

[6] *http://axis.apache.org/axis/*
[7] *http://fusesource.com/documentation/fuse-service-framework-documentation/*

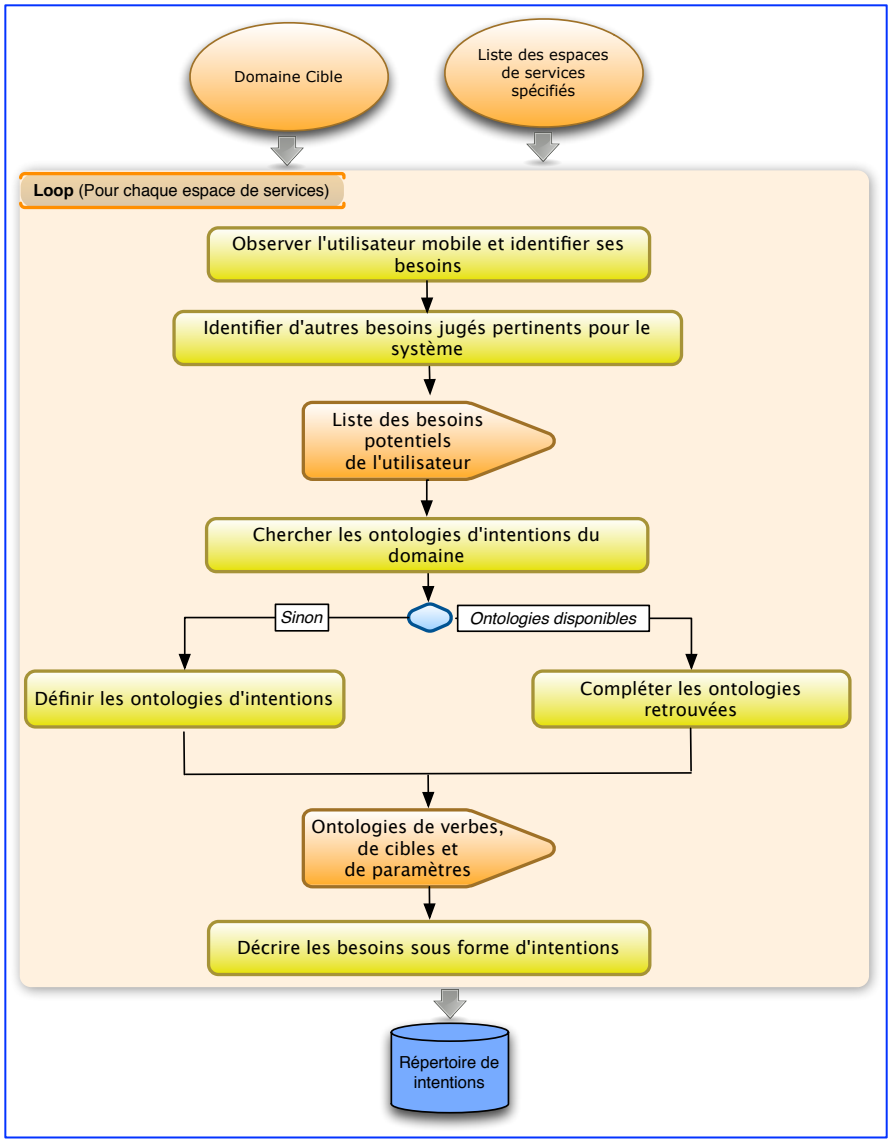


Figure 93. Processus de spécification des intentions potentielles de l'utilisateur

Le concepteur du SIP, comme l'illustre la Figure 93, doit commencer d'abord par comprendre comment l'utilisateur se sert de son SI et comment il effectue ses tâches de tous les jours. Ceci consiste à observer l'utilisateur dans son espace de services et à identifier ses différents besoins. Une liste des besoins potentiels de cet utilisateur est alors établie. Cette liste est, par la suite, validée par le concepteur, s'il souhaite rajouter d'autres besoins qu'il juge pertinentes ou simplement réalisables par le système (en partant du bas niveau).

Par la suite, ces besoins doivent être décrits sous une forme intentionnelle. Le concepteur doit représenter ces besoins sous forme d'intention définie par un verbe, qui caractérise son action, une cible, sur laquelle l'action agit, et un ensemble optionnel de paramètres. Cette démarche consiste à définir, sémantiquement et au préalable, chacun de ces éléments. Ainsi, avant de représenter les différentes intentions, il faut définir l'ontologie de verbes, l'ontologie de cibles et les ontologies spécifiant chacun des paramètres acceptés par le SIP. Ces

ontologies sont définies en fonction du domaine cible identifié au préalable. Il s'agit d'ontologies directement liées au domaine métier dans lequel s'insère le SI. Nous supposons ainsi l'existence de ces ontologies d'intentions, qui peuvent être adoptés et enrichies par le concepteur en cas de besoin. Dans le cas contraire, le concepteur est amené à définir ses propres ontologies d'intentions selon le domaine d'application. Pour la création de ces ontologies, le concepteur peut utiliser l'éditeur d'ontologie *protégé*.

A l'aide de ces ontologies et en se basant sur la liste des besoins identifiés au départ, le concepteur peut finalement décrire les besoins de l'utilisateur sous forme d'intentions. Le résultat de cette étape est un répertoire d'intentions contenant des fichiers XML décrivant les différentes intentions identifiées et décrites selon le modèle de (Prat, 1997) (*cf.* Chapitre 6).

10.3.3.2. Etape 3.2 : identification des éléments de contexte nécessaires

Cette étape d'identification des éléments de contexte nécessaires consiste à identifier les éléments de contexte observables et à établir le processus d'acquisition de contexte. Ceci débute par le choix des éléments de contexte à observer pour décrire le contexte courant de l'utilisateur ainsi que le contexte du service et du capteur lui même. De même que le contexte courant de l'utilisateur, nous rappelons que dans le cadre de notre *espace de services* (c*f.* Chapitre 5), les services et les capteurs représentent des entités actives et passives de l'espace qui sont définies en fonction de leur *contexte* Cx (et du contexte requis CxR pour les services). Ainsi, lors de cette phase nous allons identifier les éléments de contexte nécessaires pour observer l'utilisateur, le service et le capteur.

Ceci se base sur les capteurs disponibles ou susceptibles d'êtres utilisés en fonction des technologies mises à disposition. En effet, il existe différentes technologies capables de capturer un même élément de contexte. Le dilemme ici est de choisir le capteur le plus performant en termes de rapport qualité-prix, *i.e.* déterminer les technologies de capture des éléments qui seront utilisés, en tenant compte de leur intérêt pour le système et de leur coût d'application (Kirsch-Pinheiro, 2006). Chaque capteur fournit un ensemble d'informations contextuelles correspondant aux *valeurs observées* pour des *éléments de contexte* relatifs à un *sujet* bien défini. Le choix des sujets et des éléments de contexte dépend à la fois des technologies utilisées pour les observer et des moyens mis en place pour les capturer, et de l'intérêt de ces éléments pour le SIP. Le sujet observé correspond le plus souvent à l'utilisateur final, mais il n'est pas exclut d'observer d'autres éléments capables d'aider celui-ci dans son interaction avec le SIP (*e.g.* un dispositif ou une ressource). Avant toute observation d'un sujet, la nature de celui-ci doit d'abord être identifiée dans une ontologie de domaine. De la même façon, les éléments de contexte doivent être identifiés au préalable afin d'éviter l'observation d'éléments inconnus qui ne seront pas interprétés.

Le point d'entrée de cette quatrième étape est, d'une part, le domaine cible et l'ensemble des espaces de services spécifiés lors de la première étape, et d'autre part, une ontologie

multi-niveaux de contexte (*cf.* section 6.4.1) et un modèle du profil de contexte (*cf.* section 7.2.2.4.2). L'ontologie multi-niveaux de contexte représente une ontologie à deux niveaux : un *niveau supérieur* représentant les sujets et les éléments de contexte qui sont généraux et indépendant du domaine, et un *niveau inférieur* représentant les sujets et les éléments de contexte qui sont spécifiques à un domaine donné. Dans le Chapitre 6, nous représentons cette ontologie en spécifiant le premier niveau générique. Le niveau spécifique est enrichi et complété par le concepteur du SIP. Le modèle du profil de contexte (*cf.* section 7.2.2.4.2) représente un modèle qui associe un profil pour chaque sujet de contexte identifié par le concepteur. Le profil représente un ensemble d'éléments de contexte auxquels il spécifie un poids représentant son importance selon le domaine et les préférences des utilisateurs.

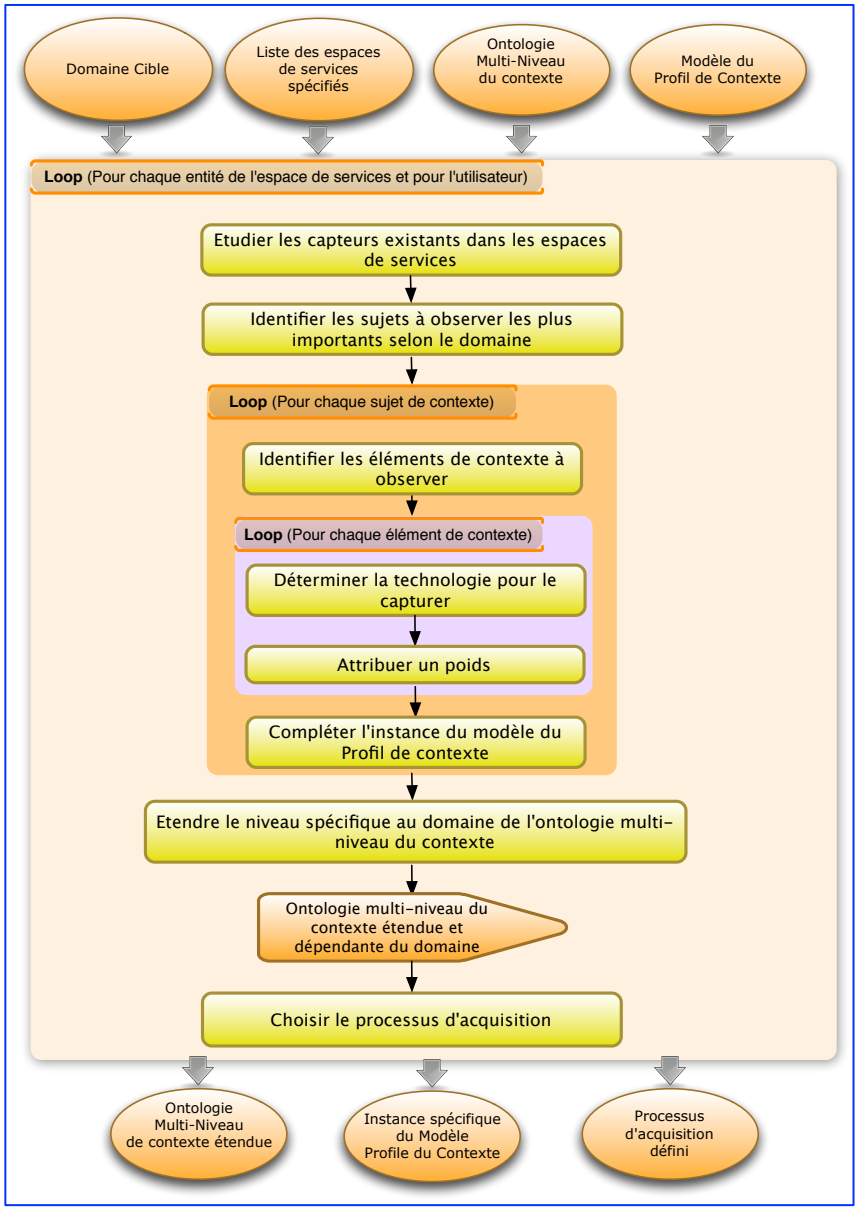


Figure 94. Processus d'identification des types de contexte nécessaires

Dans cette étape, et comme l'illustre la Figure 94, le concepteur commence donc par étudier les différents capteurs existants dans chaque espace de services spécifié, et les différents sujets à observer, considérés comme pertinents pour le SIP. Ces sujets sont forcément en rapport avec l'activité du SIP. Ensuite, pour chaque sujet déterminé, le concepteur doit déterminer les éléments de contexte qui peuvent être observés. Les éléments de contexte, eux, peuvent renseigner le SIP avec des informations complémentaires nécessaires ou intéressantes pour l'adaptation des services aux utilisateurs. Pour chaque élément de contexte identifié, le concepteur doit déterminer, d'une part, la technologie permettant de le capturer, et d'autre part, le poids qu'il souhaite attribuer à cet élément et qui représente son importance. Suite à cette étape, le concepteur commence l'instanciation du modèle du profil de contexte en l'enrichissant par ses propres informations contextuelles et les poids attribués pour chacune.

Grâce aux résultats de cette étude, le concepteur peut par la suite enrichir et étendre le niveau spécifique de l'ontologie multi-niveaux de contexte avec les éléments identifiés. En d'autres termes, il peut adopter certains éléments de l'ontologie déjà représentés, ceux qui seront effectivement utilisés dans le système en conception, et rajouter d'autres éléments qui ne figurent pas dans le modèle qu'on propose. Ceci se fait grâce à l'ontologie de contexte décrite en haut.

Après l'identification des éléments de contexte à observer, le concepteur doit spécifier le processus d'acquisition de contexte. De nombreux travaux (Gu et al., 2004) (Baldauf et al., 2007) (IST-MUSIC, 2010) ont contribué au développement de nombreux processus d'acquisition de contexte. Le concepteur est amené donc à choisir parmi cette panoplie d'architecture d'acquisition de contexte, celle qui lui convient au mieux. SOCAM (*Service-Oriented Context-Aware Middleware*) (Gu et al., 2004) et MUSIC (IST-MUSIC, 2010), par exemple, propose des *frameworks* d'acquisition de contexte qui présentent des techniques intéressantes de capture, interprétation et modélisation de contexte. Dans le cadre de notre architecture de gestionnaire de services (*cf.* Chapitre 9), le *module de gestion de contexte* représente une infrastructure pour l'acquisition de contexte de sources hétérogènes. Ce module se charge aussi de l'interprétation des informations contextuelles brutes et de leur modélisation sémantique en se basant sur le modèle de contexte (*cf.* section 6.4.1). La méthode d'acquisition de contexte est très importante lors de la phase de conception, car elle prédéfinit le style architectural au moins dans une certaine mesure (Baldauf et al., 2007).

10.3.4. Etape 4 : Description Sémantique des Services selon le contexte et l'intention

Lors de la dernière étape de description des services sémantiques sensibles au contexte et intentionnels de notre démarche méthodologique, les différents services permettant la réalisation des intentions sont décrits de manière sémantique, à l'aide du descripteur contextuel et intentionnel défini dans le Chapitre 6. Ce descripteur considère qu'un service est

proposé afin de satisfaire un ensemble supposé d'intentions attribuées aux utilisateurs potentiels de l'espace de services, dans un contexte donné.

A ce stade, le concepteur a déjà défini les différents espaces de services à prendre en compte (étape 2). Il possède également un *répertoire de services décrits en WSDL* (étape 1), un *répertoire d'intentions* avec les *ontologies d'intentions* nécessaires (étape 3), et une *ontologie* et un *modèle de contexte* (étape 4).

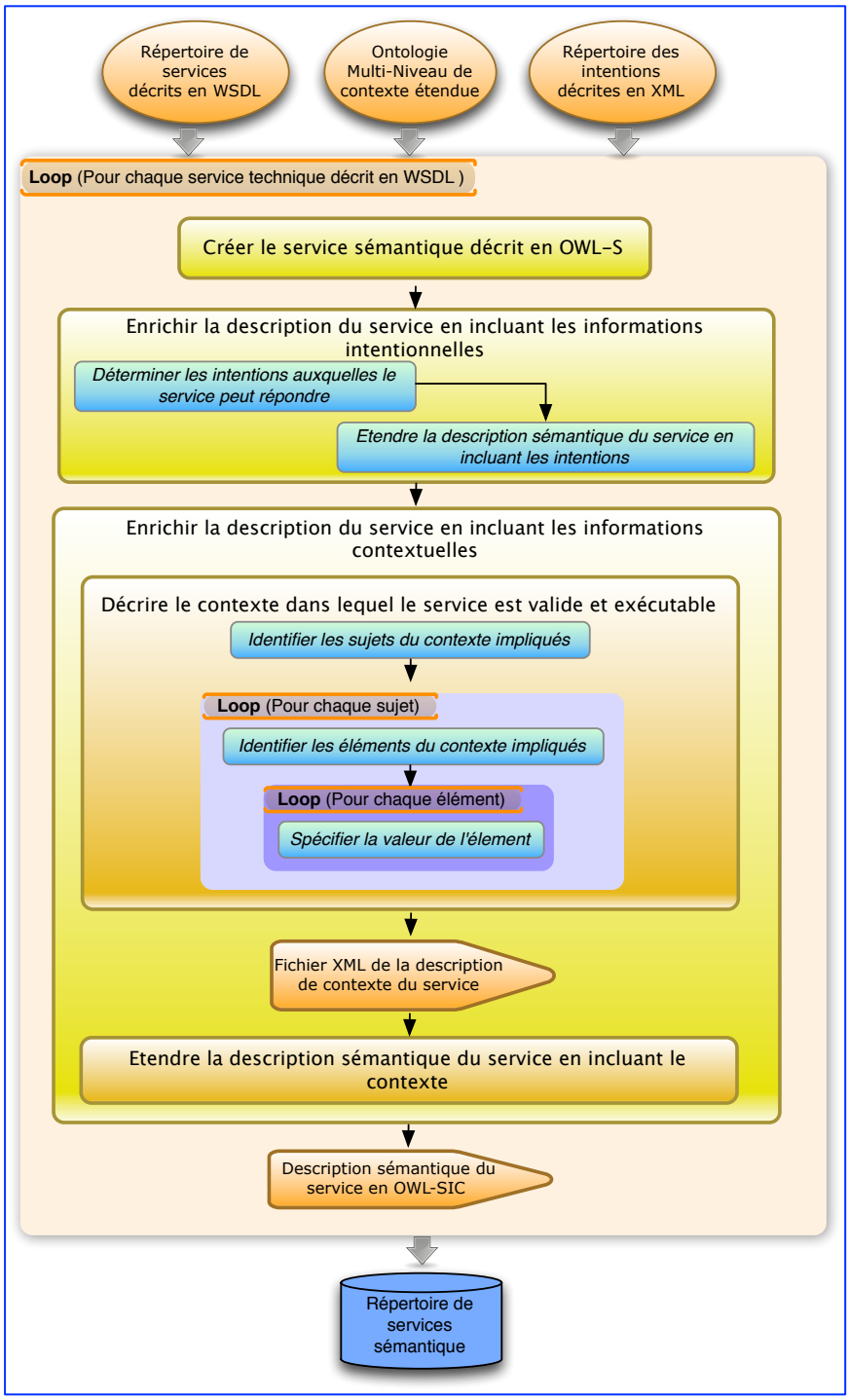


Figure 95. Processus de description des services selon l'intention et le contexte

L'étape 5 démarre, comme l'illustre la Figure 95, par la création des descriptions fonctionnelles OWL-S (Martin et al., 2007) des services à partir des descriptions WSDL. Ceci consiste en l'intégration de la technologie du Web sémantique dans la description des services afin d'atteindre l'objectif d'interopérabilité. Pour ce faire, le concepteur possède un ensemble d'outils et éditeurs capable de générer automatiquement un fichier OWL-S d'un fichier WSDL. OWL-S editor (Elenius et al., 2005) et WSDL2OWL-S (Semwebcentral, 2005), par exemple, acceptent des fichiers WSDL en entrée, extraient les informations partielles de celle-ci et génèrent automatiquement une description de base décrite en OWL-S.

Ensuite, le concepteur doit étendre cette description sémantique de service, telle que définie dans le Chapitre 6, en incluant les informations concernant les intentions et la description contextuelle caractérisant ce service pour chaque espace de services défini. Chaque service sémantique satisfait une ou plusieurs intentions. Le concepteur doit déterminer, parmi la liste des intentions prédéfinies (étape 3), celles que le service permet de satisfaire. Les intentions identifiées doivent, par la suite, être incorporées dans la description du service. Pour cela, le concepteur rajoute, dans la sous-ontologie « *service intention* » la description de service en OWL-S, la description des intentions selon le modèle de Prat (Prat, 1997). Le résultat de cette étape est un service intentionnel décrit sémantiquement.

Ensuite, le concepteur doit déterminer le contexte dans lequel ce service est exécutable et valide. D'une part, le contexte du service décrit le contexte requis, représentant les conditions contextuelles dans lesquelles le service est le plus apte à atteindre ses objectifs, ce qui correspond à des condition d'usages (*service context condition*). D'autre part, il décrit les conditions dans lesquelles le service est exécuté par le fournisseur, ce que nous appelons *« service context state »*. A partir de l'ontologie de contexte, le concepteur identifie les sujets impliqués dans la description contextuelle du service et décrit ensuite les éléments de contexte observables. Pour chaque élément de contexte, le concepteur doit spécifier les conditions selon lesquelles le service est valide et exécutable. Le concepteur est amené à spécifier la valeur exacte de cet élément ou sa marge de valeur. A la fin de ce processus, un fichier XML contenant la description contextuelle du service est créé. Cette description contextuelle du service doit être reliée à la description OWL-S du service, à travers d'attribut « *context* » attaché à la partie *« service profile »*. Cet attribut représente un pointeur URL vers le fichier de description de service.

Ainsi, chaque service décrit de cette façon représente une réalisation possible d'une ou plusieurs intentions dans un contexte donné. Par la suite, chaque description de service, décrite intentionnellement et contextuellement, est enregistrée dans un répertoire étendu de services sémantiques, lequel va être utilisé pour la découverte de services (*cf.* Chapitre 9).

Le résultat final de cette démarche méthodologique ne représente pas une description de l'espace de services, mais un ensemble de descriptions des services qu'il intègre. Ceci permet de renforcer le caractère dynamique souhaité de notre espace de services, afin d'évoluer dans le temps avec d'autres services qui se rajoutent ou d'autres qui disparaissent.

10.4. CAS D'ETUDE SECURITE ET ACCES AU SI POUR DES EMPLOYES MOBILES

Dans la perspective d'aider les concepteurs des SIP, nous illustrons dans cette section l'usage de notre démarche méthodologique. Nous appliquons les différentes étapes de cette démarche sur un cas d'étude spécifique dans le domaine du réseau mobile.

10.4.1. Introduction du cas d'étude

RésoMob est une société spécialisée en réseau mobile. Cette entreprise vend son produit à plusieurs clients et spécialement aux concessionnaires automobiles. Les employés de l'entreprise, entre autres les commerciaux et les techniciens, sont de plus en plus mobiles, travaillant chez eux, chez les clients, etc. Ils se déplacent auprès de leurs clients afin de présenter, vendre et maintenir leurs produits. Cette mobilité engendre différents espaces de travail avec différents dispositifs, différents réseaux, différentes configurations, différents profils, etc. En d'autres termes, le contexte dans lequel ces utilisateurs accèdent au SI de l'entreprise varie beaucoup en fonction de l'espace dans lequel ils se trouvent. Ainsi, la question qui en découle ici est : '*Comment gérer tous ces espaces de travail tout en permettant un accès transparent au SI* ?'

Aujourd'hui, la DSI doit prendre en considération et assister la mobilité des employés dans leurs différents espaces de travail. Ceci représente un véritable défi puisque bien souvent la DSI n'accorde pas encore son entière confiance aux nouvelles technologies, aux nouveaux dispositifs utilisés, etc. De plus, un SI n'est pas épargné des éventuels risques liés aux utilisateurs et aux failles de sécurité, qui demeurent hélas une réalité. Par exemple, la DSI doit faire face aux risques comme : l'accès par des réseaux non sécurisés, l'exposition des données clientèles, l'accès non-autorisés, etc. La DSI doit ainsi relever le défi d'assurer la mobilité des employés tout en gardant un contrôle sur le SI.

Face aux nouvelles technologies et à la mobilité de ses employés, l'entreprise a besoin d'instaurer un véritable Systèmes d'Information Pervasifs capable de garantir un accès transparent aux différents services qui sont proposés aux utilisateurs, où qu'ils soient, quelque soit leur mode d'accès et à n'importe quel moment de la journée. D'une part, ce SIP doit permettre de satisfaire les utilisateurs (les employés) en répondant à leurs besoins et en améliorant leur productivité. D'autre part, il doit permettre à la DSI de garder le contrôle sur l'accès aux différents services proposés par le système.

Concrètement, la DSI aimerait pouvoir prévoir et établir à l'avance une liste de situations possibles dans lesquelles les employés peuvent s'y retrouver. Ceci lui permettrait de décrire les services qu'elle devrait rendre disponibles tout en prenant en considération ces situations. Ainsi, un service répondant à un même besoin pourrait être décrit de différentes manières selon la situation dans laquelle il se présente. Par exemple, si un employé, qui se connecte chez le client à partir d'un réseau différent de celui de l'entreprise, demande un service bien déterminé, alors on lui proposera un service qui est certainement différent de celui proposé à

l'employé demandant le même à partir de son bureau. La différence se trouve en termes de mise en œuvre. Pour répondre à un même besoin, différentes mises en œuvre d'un service sont possibles. Par exemple, la composition utilisée pour la mise en œuvre du service peut varier en fonction du contexte dans lequel il est invoqué. En établissant ces descriptions de services correspondant aux multiples mises en œuvre, le DSI peut garder le contrôle sur le SI, tout en assurant la mobilité des employés.

10.4.2. Conception de l'espace de services

Dans cette section, nous illustrons les différentes étapes de notre démarche méthodologique en l'appliquant sur le cas d'étude décrit ci-dessus.

10.4.2.1. Spécification des espaces de services

En observant une catégorie d'utilisateurs durant leur travail et en prenant en compte leur rôle et leur mobilité, nous détectons quatre principaux espaces de services. Ces espaces représentent l'utilisateur quand il est à l'entreprise, chez lui, chez le client, à l'extérieur. On définit les espaces de services comme l'illustre le Tableau 9 :

Tableau 9. Résultat de la spécification des espaces de services

	Espace de services	**Description Espace de services**
Espace 1	*Entreprise*	Cet espace représente l'utilisateur dans son lieu de travail
Espace 2	*Domicile*	Cet espace représente l'utilisateur travaillant chez lui
Espace 3	*Chez Client*	Cet espace représente l'utilisateur quand il est chez le client
Espace 4	*Endroit public*	Cet espace représente l'utilisateur dans d'autres endroits (en dehors de l'entreprise, de son domicile ou de chez client)

10.4.2.2. Identification des fonctionnalités techniques

Pour ce scénario, nous envisageons que les services techniques correspondants aux fonctionnalités existantes du système d'information sont disponibles.

Le domaine de ce scénario est le domaine du réseau et de l'infrastructure pour une entreprise de réseau mobile. Dans ce domaine, le Système d'Information existant offre des fonctionnalités telles que l'accès à la fiche client, l'édition des propositions, la consultation des commandes, l'organisation de réunions virtuelles, etc. Dans le cadre de notre SIP, et pour chacune de ces fonctionnalités, nous déterminons les services techniques correspondants.

Le Tableau 10 décrit les fonctionnalités et les services techniques identifiés.

Tableau 10. Résultat de l'étape d'identification des fonctionnalités techniques

Domaine Cible	Fonctionnalités Pertinentes	Services techniques correspondants
Infrastructure et Réseau	*Consultation Fiche Client Avec VPN*	**AccessClientViewVPN_Service**
	Consultation Fiche Client Avec SSL	**AccessClientViewSSL_Service**
	Consultation Fiche Client Avec Cryptage	**AccessClientViewEncryption_Service**
	Edition Proposition Avec Fax	**ProposalEditionFax_Service**
	Edition Proposition Avec Cryptage	**ProposalEditionEncryption_Service**
	Edition Proposition Avec VPN	**ProposalEditionVPN_Service**
	Consultation Commande	**ViewCommand_Service**
	Consultation Commande Avec VPN	**ViewConferenceVPN_Service**
	Organisation Video-Conférence	**RequestVideoConference_Service**
	Organisation Audio-Conférence Bas Niveau	**RequestLowQualityAudioConference_Service**
	Organisation Video-Conférence Haute Définition	**RequestHighQualityVideoConference_Service**
	Organisation Video-Conférence Avec Cryptage	**RequestEncryptedVideoConference_Service**
	Recherche Restaurant	**FindRestaurant_Service**

10.4.2.3. Identification du couple <Intention, Contexte>

Pour l'identification des couples <Intention, Contexte> des espaces de services identifiés, nous choisissons la première alternative, à savoir l'identification de ce couple à partir du haut niveau afin d'éliciter les intentions potentielles de l'utilisateur et les éléments de contexte que le concepteur est capable de trouver vis-à-vis de l'utilisateur. Ainsi, nous identifions le couple <Intention, Contexte> à partir de l'*utilisateur.*

10.4.2.3.1. Spécification des intentions potentielles de l'utilisateur

Durant cette phase, nous analysons le rôle de l'utilisateur, ainsi que son comportement au quotidien. Cette analyse permet de déterminer une première liste des besoins potentiels de

l'utilisateur qu'il souhaite voir satisfaits par son SIP. Par la suite, et à partir des fonctionnalités que le SIP va offrir à l'utilisateur (identifiées lors de la deuxième étape de cette démarche), nous spécifions une liste des besoins auxquels les fonctionnalités offertes par le SIP peuvent répondre. La liste finale des besoins des utilisateurs qui peuvent être satisfaits par le SIP est, par la suite, déterminée en confrontant les deux listes précédemment identifiées. Cette confrontation entre les deux listes des besoins, va permettre de déterminer les besoins nécessaires pour l'utilisateur et qui sont concrètement réalisables par le SIP. Le Tableau 11 illustre la liste finale des besoins des utilisateurs.

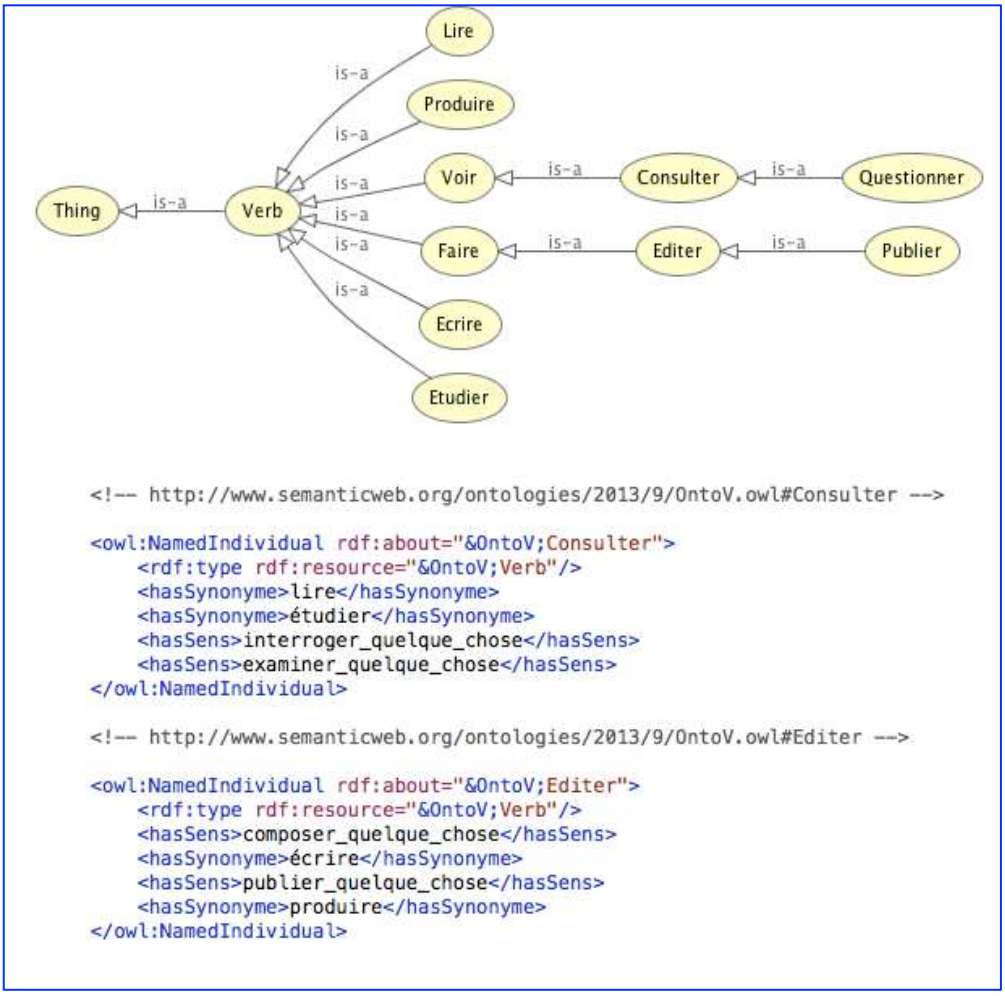
Tableau 11. Résultat de la spécification des besoins de l'utilisateur

	Besoins de l'utilisateur	**Description du besoin de l'utilisateur**
Besoin 1	*La consultation de la fiche client*	L'utilisateur a besoin de consulter la fiche technique de tous les clients de l'entreprise.
Besoin 2	*L'édition de la proposition*	L'utilisateur a besoin d'écrire et d'éditer les propositions destinées aux clients.
Besoin 3	*La consultation de la commande*	L'utilisateur a besoin de consulter l'avancement et le détail des commandes.
Besoin 4	*L'organisation de réunion*	L'utilisateur a besoin de faire des réunions virtuelles.
Besoin 5	*La recherche d'un restaurant*	L'utilisateur a besoin de trouver un restaurant agréé par son entreprise pour aller manger quand il est à l'extérieur.

Ces besoins vont par la suite être décrits sous forme d'intentions. Nous commençons par déterminer nos ontologies de verbes et de cibles (*cf.* section 6.3.1.3), nécessaires pour analyser sémantiquement ces intentions. Nous créons ces ontologies dans le domaine du réseau et de l'infrastructure.

L'ontologie de verbes est décrite en se basant sur le dictionnaire en ligne de Larousse[9]. Larousse fournit une description complète d'un verbe, en illustrant ses sens, ses synonymes, ses hypernymes, ses hyponymes, etc. Un extrait de l'ontologie de verbes, construite à partir de l'outil d'édition d'ontologie *Protégé*, est illustré à la Figure 96. Cet extrait présente une description sémantique des verbes « *consulter* » et « *éditer* », ainsi que leur relation de type « *hasSens* » et « *hasSynonym* ». Cet extrait illustre également les relations d'héritage entre les verbes qui reflètent la relation d'*hypernymie* et d'*hyponymie*.

[9] *http://www.larousse.fr/*




```
<!-- http://www.semanticweb.org/ontologies/2013/9/OntoV.owl#Consulter -->

<owl:NamedIndividual rdf:about="&OntoV;Consulter">
    <rdf:type rdf:resource="&OntoV;Verb"/>
    <hasSynonyme>lire</hasSynonyme>
    <hasSynonyme>étudier</hasSynonyme>
    <hasSens>interroger_quelque_chose</hasSens>
    <hasSens>examiner_quelque_chose</hasSens>
</owl:NamedIndividual>

<!-- http://www.semanticweb.org/ontologies/2013/9/OntoV.owl#Editer -->

<owl:NamedIndividual rdf:about="&OntoV;Editer">
    <rdf:type rdf:resource="&OntoV;Verb"/>
    <hasSens>composer_quelque_chose</hasSens>
    <hasSynonyme>écrire</hasSynonyme>
    <hasSens>publier_quelque_chose</hasSens>
    <hasSynonyme>produire</hasSynonyme>
</owl:NamedIndividual>
```

Figure 96. Extrait de l'ontologie des verbes

De même, nous illustrons un extrait de l'ontologie de cibles à la Figure 97, construite également en utilisant *protégé*. Cet extrait présente une description sémantique de certaines cibles qui sont possibles dans le domaine spécifié, à savoir : *commande, réunion*, etc.

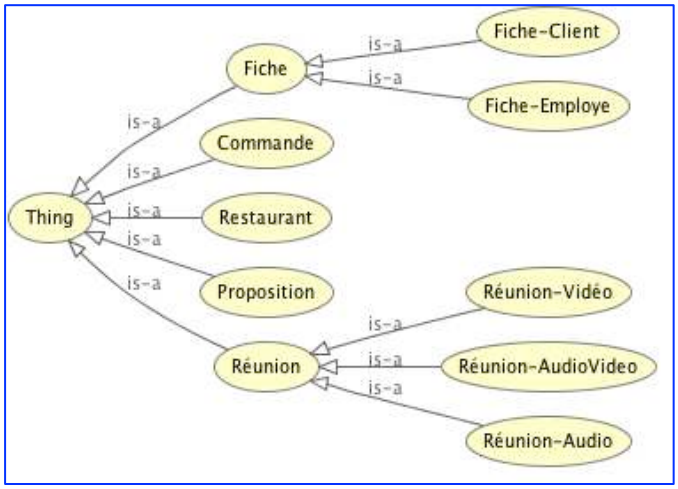


Figure 97. Extrait de l'ontologie des Cibles

A partir de la liste des besoins des utilisateurs et des ontologies de verbes et de cibles, nous décrivons ces besoins sous forme d'intentions. Nous suivons le modèle de Prat (Prat, 1997), en représentant l'intention sous forme de verbe et de cible. Ces intentions sont décrites comme l'illustre le Tableau 12 :

Tableau 12. Résultat de la spécification des intentions de l'utilisateur

	Besoin	**Intention**	**Modélisation de l'intention**
Intention 1	*La consultation de la fiche client*	*Consulter Fiche_Client*	<?xml version="1.0" encoding="UTF-8"?> <intention> <verb> ***consulter*** </verb> <target> <object> ***Fiche_Client*** </object> </target> </intention>
Intention 2	*L'édition de la proposition*	*Editer Proposition*	<?xml version="1.0" encoding="UTF-8"?> <intention> <verb> ***éditer*** </verb> <target> <result> ***proposition*** </result> </target> </intention>
Intention 3	*La consultation de la commande*	*Consulter Commande*	<?xml version="1.0" encoding="UTF-8"?> <intention> <verb> ***consulter*** </verb> <target> <object> ***commande*** </object> </target> </intention>
Intention 4	*L'organisation de réunion*	*Organiser Réunion*	<?xml version="1.0" encoding="UTF-8"?> <intention> <verb> ***organiser*** </verb> <target> <object> ***réunion*** </object> </target> </intention>
Intention 5	*La recherche de restaurant*	*Rechercher Restaurant*	<?xml version="1.0" encoding="UTF-8"?> <intention> <verb> ***rechercher*** </verb> <target> <object> ***restaurant*** </object> </target> </intention>

Nous rappelons, qu'il faudra décrire sémantiquement chaque verbe et cible dans les ontologies précédemment décrites.

10.4.2.3.2. Identification des éléments de contexte nécessaires

Lors de cette étape, nous commençons par fixer les entités de contexte que nous souhaitons observer dans le cadre des espaces de services identifiés. Ensuite, pour chaque entité de contexte spécifiée, nous identifions l'ensemble des éléments de contexte qui caractérisent cette entité et décrit l'utilisateur dans son environnement. Par la suite, en analysant les différentes fonctionnalités du SIP, nous détectons certains éléments de contexte qui peuvent caractériser le contexte dans lequel ces fonctionnalités peuvent être valides et exécutables. Après avoir analysé l'ensemble des différents éléments de contexte identifiés, nous déterminons une liste finale des entités de contexte à observer et de l'ensemble des éléments de contexte caractérisant ces entités.

Pour illustrer cette étape, nous nous concentrons sur l'utilisateur uniquement. Plus concrètement, nous déterminons deux entités de contexte qui sont les plus pertinentes à observer dans les espaces de services de l'utilisateur, à savoir : l'*utilisateur* et le *dispositif*. Par la suite, pour chacune de ces entités de contexte, nous déterminons les éléments de contexte à collecter, comme l'illustre le Tableau 13.

Tableau 13. Résultat de la spécification des éléments et sujets de contexte nécessaires

Entité de Contexte	Elément de Contexte	Nature de L'élément de Contexte	Capteur de Contexte
Utilisateur	L'**âge** de l'utilisateur	***Statique***	-
	Le ***rôle*** de l'utilisateur	***Statique***	-
	La ***localisation*** de l'utilisateur représentée par les coordonnées ***GPS***	***Dynamique***	*capteur_GPS*
	L'***expertise*** de l'utilisateur	***Statique***	-
Dispositif	Le ***type*** du dispositif	***Dynamique***	*capteur_Type_Dispositif*
	La ***mémoire*** du dispositif	***Dynamique***	*capteur_Mémoire_Dispositif*
	Le ***type du réseau***	***Dynamique***	*capteur_Type_Réseau*
	Le **nom *du Réseau***	***Dynamique***	*capteur_ssid_Réseau*

Suite à l'identification des sujets et des éléments de contexte, nous complètons l'ontologie multi-niveaux de contexte, que nous avons présentée à la section 6.4.1, par les éléments et les sujets que nous jugeons pertinents dans le domaine cible. Par exemple, nous rajoutons l'« *expertise* » de l'utilisateur comme une spécialisation de l'élément de contexte « *profil* ». Cet élément de contexte reflète le niveau d'expertise en technique réseau de l'utilisateur dans le domaine Informatique. De plus, nous spécialisons le sujet de contexte « dispositif » en

« dispositif de communication », « ordinateur » et « dispositif mobile ». Un extrait de notre extension de l'ontologie multi-niveaux de contexte est illustré à la Figure 98.

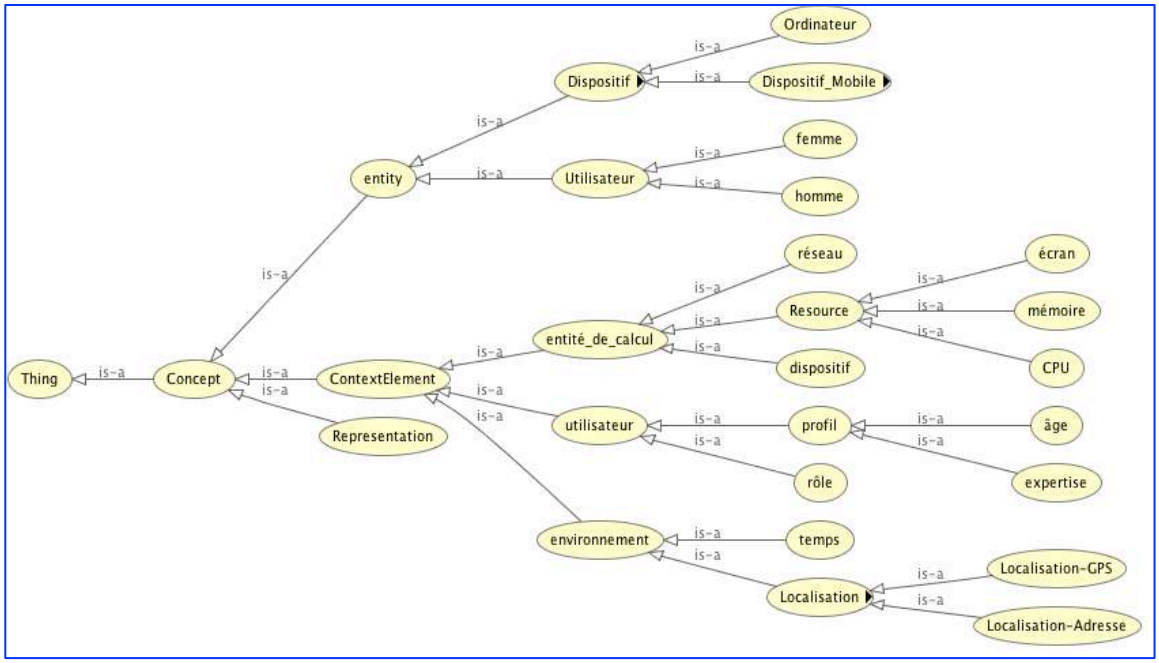


Figure 98. Extrait de l'ontologie multi-niveaux de contexte

Pour ces éléments de contexte, nous attribuons des poids à chacun d'entre eux, formant ainsi le modèle de profil de contexte (*cf.* section 7.2.2.4.2). Nous fixons ces poids, selon l'importance de chaque élément de contexte dans le domaine. La Figure 99 illustre un extrait de notre modèle de profil de contexte. Selon cet extrait, la *localisation* a un poids égal à *1*. Ceci reflète l'importance que joue cet élément de contexte pour l'utilisateur. Par contre, l'élément de contexte *mémoire* joue un rôle moins important selon l'utilisateur, nous lui avons attribué un poids égal à 0,6. Ce poids indique que l'élément de contexte mémoire jouera un rôle moins influent lors des processus de découverte et de prédiction de services.

```
<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/2011/10/ProfilContextModel.owl#">

    <!-- http://www.semanticweb.org/ontologies/2011/10/ProfilContextModel.owl#Profil_Attribut_GPS -->

    <owl:NamedIndividual rdf:about="&ProfilContextModel;Profil_Attribut_GPS">
        <rdf:type rdf:resource="&ProfilContextModel;Profil-Attribut"/>
        <hasWeight rdf:datatype="&xsd;double">0.8</hasWeight>
        <hasAttribut rdf:resource="&ProfilContextModel;GPS"/>
        <hasProfile rdf:resource="&ProfilContextModel;Profil1"/>
    </owl:NamedIndividual>

    <!-- http://www.semanticweb.org/ontologies/2011/10/ProfilContextModel.owl#Profil_Attribut_Memoire -->

    <owl:NamedIndividual rdf:about="&ProfilContextModel;Profile_Attribut_Mémoire">
        <rdf:type rdf:resource="&ProfilContextModel;Profil-Attribut"/>
        <hasWeight rdf:datatype="&xsd;double">0.6</hasWeight>
        <hasAttribut rdf:resource="&ProfilContextModel;Mémoire"/>
        <hasProfile rdf:resource="&ProfilContextModel;Profil2"/>
    </owl:NamedIndividual>

</rdf:RDF>
```

Figure 99. Extrait du modèle de profil de contexte

A ce stade, nous avons spécifié nos espaces de services. Nous avons également identifié les entités passives (capteurs) qui vont être intégrées aux espaces de services, ainsi que les éléments nécessaires, à savoir les intentions potentielles et les sujets et les éléments de contexte, pour décrire nos services selon une vision intentionnelle et contextuelle. Nous illustrons notre dernière étape de la démarche méthodologique dans la section suivante.

10.4.2.4. Description sémantique des services

Dans cette section, nous présentons notre description des services offerts par le SIP que nous souhaitons conceptualiser à travers les différents espaces de services identifiés. A partir des descriptions de services techniques identifiés lors de la deuxième étape de notre démarche, nous décrivons sémantiquement ces services en rajoutant l'intention que chacun d'entre eux est capable de satisfaire, ainsi que le contexte dans le quel ce service est valide et exécutable. Le Tableau 14 illustre notre description des services.

Tableau 14. Extrait de la description intentionnelle et contextuelle des services (*cf.* Annexe B.1)

Service	*Intention*	*Contexte requis*	*Actions réalisées par le service*
Service 1	I1 Consulter fiche client	- Dispositif.Réseau.type ≠ Ethernet - Utilisateur.Localisation.Nom ≠ Entreprise - Utilisateur.Rôle = Commercial - Utilisateur.Profil.Expertise = Faible	- ***AccessClientViewVPN*** (TunnelVPN, SSLAuthentification, DataEncryption, ClientListPage, ViewClientWS)
Service 2	I1 Consulter fiche client	- Dispositif.Réseau.type≠ Ethernet - Utilisateur.Profil.Expertise = Elevé - Utilisateur.Rôle = Commercial	- ***AccessClientView*** (SSLAuthentificationWS, FindClientByAddressWS, DetailedClientView)
Service 4	I2 Etablir proposition	- Dispositif.Réseau.type = Ethernet - Utilisateur.Localisation.Nom = Entreprise - Dispositif.Mémoire > 512	- ***ProposalEditionFax*** (ProposalEditionWS, FaxService)
Service 6	I2 Etablir proposition	- Dispositif.Réseau.Type ≠ Ethernet - Dispositif.Type ≠ Iphone - Dispositif.Mémoire >= 1024 - Utilisateur.Profil.Expertise = élevé - Utilisateur.Localisation.Nom ≠ Entreprise	- ***ProposalEditionEmail*** (TunnelVPN, AuthentificationWS, ProposalEditionWS, EmailClient)
Service 8	I3 Consulter commande	- Utilisateur.Rôle = Commercial - Dispositif.Réseau.type ≠ Ethernet - Utilisateur.Localisation.Nom ≠ Entreprise - Utilisateur.Profil.Expertise = Low	- ***ViewCommandVPN*** (TunnelVPN, AuthentificationWS, FindClientByAddressWS, TraceClientCommand°
Service 12	I4 Réaliser réunion	- Utilisateur.Rôle = Commercial - Utilisateur.Localisation.Nom ≠ Entreprise - Dispositif.Mémoire > 512	- ***RequestVideoConference*** (RequestVirtualRoom, RequestBordWidth,

		- Dispositif.Réseau.Type ≠ Ethernet	StartVideoConfWS)
Service 14	I4 Réaliser réunion	- Utilisateur.Rôle = Commercial - Utilisateur.Localisation.Nom ≠ Entreprise - Dispositif.Mémoire > 512 - Dispositif.Réseau.type ≠ Ethernet - Dispositif.Type = Android	- ***RequestEncryptedLow QualityVideo Conference*** (RequestVirtualRoom, DataEncryption, StartLowQualityVideoConfWS)
Service 17	I5 Rechercher Restaurant	- Dispositif.Réseau.type ≠ Ethernet - Utilisateur.Rôle = Commercial - Utilisateur.Localisation.Nom ≠ Entreprise	***-FindRestaurantVPN*** (TunnelVPN, SSLAuthentification, FindNearestRestaurant, RestaurantList,ViewRestaurant)

A ce stade nous avons déterminé l'ensemble des services intentionnels et contextuels qui s'intègrent dans les différents espaces de services que nous avons spécifiés dans la section 10.4.2.1. La Figure 100 illustre ces différents services avec les intentions qu'ils satisfont, lesquelles s'intègrent ainsi dans de multiple espaces de services, reflétant ainsi la nature perméable de notre notion d'espaces de services (*cf.* section 5.4.3). Dans les sections suivantes, nous présentons et analysons plus en détails notre cas d'étude.

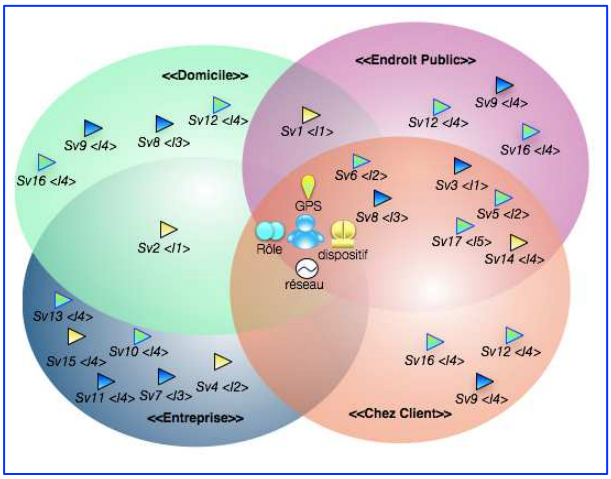


Figure 100. Les services intentionnels et contextuels intégrés dans l'espace de services

Comme l'illustre la Figure 100, une même intention dans le même espace de services peut être satisfaite de manière différente selon le contexte (plusieurs services pour une même intention). De plus, une même intention (*e.g* l'intention *I4*) peut être identifiée dans plusieurs espaces de services.

Nous présentons par la suite un cas d'étude illustrant la satisfaction des intentions de l'utilisateur par différentes réalisations de services en fonction de son contexte courant.

10.4.3. Description du cas d'étude

Jean est le commercial de l'entreprise. Il travaille chez lui habituellement le jeudi matin. Il accède à son SI via sa tablette Android et commence à écrire une proposition qu'il doit présenter en fin d'après midi à un client. Ce jour là, Jean accède via un réseau différent de celui de l'entreprise (celui de sa maison). Différentes implémentations existent pour satisfaire son besoin, mais il ne comprend pas comment le faire. Il n'a ni la compétence ni l'expertise technique en réseau nécessaire pour ouvrir un tunnel VPN et régler la configuration. Il a besoin d'un autre moyen qui lui permette de répondre à son besoin d'une façon transparente et sans être amené à faire un choix parmi les implémentations présentées. Ainsi le SIP vérifie dans le profil utilisateur qu'il n'a pas une forte expertise en réseau, que le réseau qu'il utilise est différent de celui de l'entreprise, que son type de dispositif est une tablette Android 4.1 et que la mémoire de son dispositif est de 2048 Mo (donc supérieure à 1024 Mo). Ces informations observées du contexte utilisateur permettront au système de sélectionner le service qui s'adapte au mieux à ce contexte et à l'intention de Jean d'écrire une proposition. A partir de ces informations, le système va choisir d'exécuter un service composite qui va ouvrir un tunnel VPN, déclencher le service Web d'authentification et finalement exécuter le service d'édition de proposition et le service d'envoi de mail. Jean ne sait pas ce que signifie un tunnel VPN ou à quoi ça sert. Il ne sait pas comment ça fonctionne et quand faut il l'utiliser. Si Jean devrait faire seul le choix de la mise en œuvre ou de la composition du service qu'il lui faut, vu son contexte, il en aurait été incapable. Pour l'aider, tous les détails techniques doivent rester cachés et que l'accès aux fonctionnalités dont il a besoin soit transparent. La seule chose qui compte aux yeux de Jean est son intention d'écrire une proposition. Pour lui, le système doit l'aider à la satisfaire quelque soit le moyen utilisé *(Situation 1)*.

En fin de matinée, Jean passe à l'entreprise pour réaliser une réunion avec son directeur. Bien installé dans son bureau, il utilise son PC *Intel Core 2 Duo* de 1024 de mémoire. A l'entreprise, Jean est connecté au réseau Ethernet de l'entreprise. Il accède directement au SI et demande de réaliser une réunion. Puisque la salle de réunion n'est pas libre à cet instant, le système lui propose une réunion virtuelle et lance le service de réunion en haute définition *(Situation 2)*. De son côté, le directeur est sur un autre site de l'entreprise, connecté via son PC *Intel Core 2 Duo* de 2 Go de mémoire. Le système, en comparant son contexte avec celui d'usage des services disponibles, va décider d'exécuter un service qui lui permet de joindre la réunion virtuelle et lance également la vidéo conférence en haute définition *(Situation 2 bis)*.

A la fin de la réunion, il est midi, Jean part déjeuner avec son collègue Pierre. Il se rappelle qu'il doit modifier certains points de sa proposition, selon la réunion du matin, avant de partir chez le client. Jean demande l'aide de son collègue Pierre. Pierre alors utilise son IPad, se connecte via le réseau 3G et accède au système dans le but d'éditer la proposition de Jean et de l'envoyer par mail. Le système constate que Pierre se connecte de l'extérieur via un réseau différent de celui de l'entreprise. De plus, contrairement à Jean, Pierre a une très bonne expertise, alors le système lui choisit un autre service pour l'édition de la proposition. Celui-

ci, après avoir permis l'édition de la proposition, déclenche le service de messagerie, utilisant le service de cryptage de données, étant donné que Pierre est à l'extérieur *(Situation 3)*.

En arrivant chez le client, Jean se connecte via le réseau WI-FI. Il commence la présentation du produit au client. Au milieu de la présentation, suite aux multiples questions du client, Jean a besoin d'un technicien de l'équipe afin qu'il puisse expliquer au client certains détails. Il appelle alors Louis le technicien qui se trouve dans l'entreprise. D'un côté, Jean accède au SI et demande le service assurant une conférence avec son collègue. Le système a compris son intention, et observe alors son contexte d'utilisation afin de répondre au mieux à son intention. Le système observe que Jean se trouve chez le client, se connectant via un réseau différent de celui de l'entreprise à travers une tablette Android dont la capacité de mémoire est de 768 Mo et que Jean a le rôle de commercial. A partir de ces informations, le système propose à Jean un service qui se compose d'un service vidéo conférence en bas débit et du service de cryptage des données *(Situation 4)*. De l'autre côté, Louis demande le même service mais dans un autre contexte : dans l'entreprise, se connectant via le réseau de l'entreprise et jouant son rôle de technicien. Le système reconnait son intention, qui est la même que Jean, mais dans un contexte différent. Il lui propose alors un service qui se compose d'un service permettant de joindre une réunion virtuelle et d'un service d'audio conférence en bas débit *(Situation 4 bis)*. Au cours de la démonstration, Jean doit accéder à la fiche de son client. Puisqu'il est chez le client, le système propose à Jean d'exécuter un service composé du service Web d'authentification SSL et du service d'accès à la fiche client, le tout en accédant via un Tunnel VPN et en cryptant les données pour des raisons de sécurité *(Situation 5)*. Ensuite, Jean demande à Louis de vérifier certaines informations sur le client. Alors, Louis demande d'accéder à la fiche du client. Puisqu'il est dans l'entreprise et qu'il a un niveau élevé d'expertise, le système lui propose d'exécuter un service composé du service Web d'authentification SSL et du service d'accès à la fiche client *(Situation 6)*.

Quand Jean termine son travail chez le client, il a faim. Alors il décide d'aller déjeuner dans le coin. Il demande à son SI de lui chercher le restaurant le plus proche et qui est agréé par son entreprise. Etant donné que Jean est à l'extérieur, alors le système lui propose d'exécuter un service composé d'un service de recherche du restaurant le plus proche avec une connexion au tunnel VPN et une authentification SSL *(Situation 7)*.

10.4.3.1. Description des situations de l'utilisateur

Nous présentons, dans le Tableau 15, l'ensemble des situations des utilisateurs décrites dans la section précédente. La *situation* est composée d'une part, de l'*intention* de l'utilisateur dans un *contexte* observé, et d'autre part de la *mise en œuvre du service* choisie avec le *contexte* dans lequel ce service est valide et exécutable. Du côté utilisateur, l'intention exprime le besoin de l'utilisateur. Le contexte représente les différentes valeurs capturées pour chaque attribut des éléments de contexte observé. Du côté service, la mise en œuvre représente la composition de service choisie afin de satisfaire l'utilisateur. Le contexte représente une description du contexte dans lequel cette mise œuvre est valide et exécutable.

Tableau 15. Extrait de la description des situations de l'utilisateur en termes de <Intention, Contexte, Service> (*cf.* Annexe B.2)

Situation	Sujet de Contexte	Espace de Services	Utilisateur		Service sélectionné
			Intention	***Contexte***	
Situation 1	Jean	Domicile	I2 Etablir proposition	- Utilisateur.Rôle = Commercial - Utilisateur.Profil.Age = 42 - Utilisateur Profil.Expertise = faible - Utilisateur.localisation.Nom = domicile - Utilisateur.Temps = Matin - Dispositif.Type = Android 4.1 - Dispositif.Mémoire = 2048 - Dispositif.Réseau.Type = WiFi - Dispositif.Réseau.Nom = Jean-WiFi	**Service 6**
Situation 2 bis	Directeur	Entreprise	I4 Réaliser réunion	- Utilisateur.Rôle = Directeur - Utilisateur.Profil.Age = 54 - Utilisateur Profil.Expertise = High - Utilisateur.localisation.Nom = Entreprise - Utilisateur.Temps = Après-midi - Dispositif.Type = Intel Core 2 Duo - Dispositif.Mémoire = 2 Go - Dispositif.Réseau.Type = Ethernet - Dispositif.Réseau.Nom = 192.168.13.139	**Service 10**

Dans la section suivante, nous analysons ce cas d'étude par rapport à notre vision intentionnelle et contextuelle des SIP.

10.4.3.2. Analyse du scénario

Ce cas d'étude, présentant les employés mobiles d'une entreprise dans différents espaces de travail, illustre l'apport de nos travaux de recherche dans le domaine des Systèmes d'Information Pervasifs (SIP). Ce travail permet à l'utilisateur d'accéder en toute transparence à son SI quelque soit son niveau de compétence et de compréhension. Notre approche permet de cacher la complexité des systèmes en cachant au maximum les détails techniques qui demeurent difficiles à assimiler par des utilisateurs non-experts.

Avec les Systèmes d'Information existants, l'utilisateur doit avoir un certain niveau de compréhension puisqu'il est amené à comprendre comment certains services fonctionnent et quand est ce qu'il faut les utiliser. L'utilisateur, dans certain cas, est obligé de faire seul le choix de la mise en œuvre ou de la composition du service qu'il lui faut, étant donné son contexte. Afin de palier cette problématique, notre système est conçu dans la perspective de répondre aux besoins de l'utilisateur sans pour autant lui demander d'être impliqué dans le choix de l'exécution du service approprié et dans sa mise en œuvre. De plus notre SIP ne

demande pas à l'utilisateur de fournir des efforts afin de comprendre certains détails techniques nécessaires pour la réalisation de son besoin.

Comme l'illustre le Tableau 15, l'utilisateur n'est pas censé connaître toutes les réalisations possibles des services. Il n'a qu'à exprimer sa réelle intention derrière sa demande d'un service, et le SIP se charge de la satisfaire par le service le plus adéquat selon son contexte courant. Ainsi, un utilisateur qui n'a pas de fortes compétences techniques et qui n'a pas de forte expertise, ne va pas se retrouver face à toutes les implémentations possibles d'un service. Ainsi, notre objectif est de satisfaire le besoin de l'utilisateur quelque soit les moyens fournis. Ceci met en avant le caractère transparent et centré utilisateur de notre SIP. Le caractère centré utilisateur est garanti en prenant en considération les intentions des utilisateurs dans leur contexte courant. De plus, la transparence est assurée en cachant la complexité technique à Jean, qui a un niveau faible d'expertise, en lui proposant directement une connexion au tunnel VPN, par exemple. En outre, quand Jean se déplace d'un espace de services à un autre, il ne se rend pas vraiment compte, parce que le SIP se charge de répondre à ses intentions en s'adaptant directement à ce changement, facilitant ainsi sa tâche.

A partir ce cas d'étude, nous relevons deux points importants pris en compte dans notre vision intentionnelle et contextuelle des SIP. Le premier point est que pour « *pour une même intention différentes réalisations de services existent* ». Prenons, par exemple, le cas d'un commercial qui a besoin d'établir une proposition. Cette intention a été satisfaite de différentes manières. Dans un premier cas (*Situation 3*), l'intention est satisfaite par la mise en œuvre du service « *ProposalEdition* » qui est composé des services : « *Authentification WS* », « *ProposalEditionWS* », « *DataEncryption* » et « *EmailClient* ». Dans un deuxième cas (*Situation 1*), l'intention est satisfaite par la mise en œuvre du service « *ProposalEdition* » qui est composé des services : « *TunnelVPN* », « *AuthentificationWS* », « *ProposalEditionWS* » et « *EmailClient* ». D'un côté, la mise en œuvre de la *situation 3* a fait appel au service de cryptage de donnés (ce qui n'est pas le cas dans la *situation 1*). Ceci est du à l'information capturée sur la localisation de l'utilisateur et qui indique qu'il est à l'extérieur de l'entreprise. De l'autre côté, la mise en œuvre de la *situation 1* a fait appel au service Tunnel VPN (ce qui n'est pas le cas dans la situation 3). Ceci est dû au fait que, d'après le profil de l'utilisateur, son niveau d'expertise n'est pas élevé (il n'a pas les compétences nécessaires pour comprendre les détails techniques). Ainsi, on note que pour une même intention, il existe différentes réalisations possibles. Le système se base sur les informations observées du contexte de l'utilisateur qui vont être primordiales dans le choix de la mise en œuvre du service la plus appropriée et qui répond au mieux à l'intention de l'utilisateur.

Le deuxième point représente le fait que *« le contexte influence le choix de la réalisation* ». Par exemple, pour la même intention réaliser réunion, on a deux contextes d'utilisation différents. Le premier contexte (*Situation 4 bis*) présente un technicien qui a un niveau d'expertise élevé, se trouvant à l'entreprise et se connectant via son Intel Core 2 Duo à travers le réseau Ethernet. Le deuxième contexte (*Situation 4*) présente un commercial qui a un niveau d'expertise de bas niveau, se trouvant à l'extérieur de l'entreprise, se connectant via son Android 4.1 à travers le réseau Wifi public. On remarque que les deux utilisateurs ont le

même besoin mais dans des contextes différents. Par conséquent, la mise en œuvre prise pour satisfaire ce besoin est différente dans ces deux situations. Dans la *Situation 4 bis,* le système décide d'exécuter la mise en œuvre du service *« RequestConference »* qui est composé des services : « *JoinVirtualRoom* » et « *StartLowQualityAudioConfWS* ». Par contre, dans la *Situation 4*, le système décide d'exécuter la mise œuvre du service *« RequestConference »* qui est composé des services : « *JoinVirtualRoom* », *« DataEncryption »* et « *StartLowQualityVideoConfWS »*. Ainsi, on note que le contexte joue un rôle essentiel dans le choix de la réalisation la plus appropriée afin de satisfaire le besoin de l'utilisateur.

Ainsi, l'utilisateur exprime son intention dans un contexte donné. Ce contexte influence le choix de la réalisation du service qui satisfait cette intention. Ce mécanisme de sélection du service le plus approprié est géré par le processus de découverte de services guidée par le contexte et l'intention (*cf.* Chapitre 7). Les différents services, illustrés au Tableau 15*,* ont été sélectionnés par le processus de découverte de services. Par exemple, Jean exprime son intention *« réaliser réunion »*. Cette intention est envoyée à notre architecture de gestionnaire de SIP qui se charge, par la suite, d'enrichir cette intention avec le contexte courant de Jean. L'intention enrichie par le contexte est, ensuite, envoyée au module de découverte de services. Ce module compare sémantiquement l'intention et le contexte de Jean avec les intentions et les contextes des services disponibles. Il détermine que le service *« RequestConference »* réalisé par les services « *JoinVirtualRoom* », *« DataEncryption »* et « *StartLowQualityVideoConfWS »*, est le service le plus approprié.

Ensuite en observant les traces de Jean pendant une période donnée, nous avons remarqué que lorsqu'il part chez le client, il part souvent déjeuner dans le coin. Cette information représente une étape du schéma de comportement de Jean. Ainsi, le mécanisme de prédiction de services *(cf. Chapitre 8)*, nous aidera à déterminer le besoin de Jean à chercher un restaurant dans l'entourage du client sans qu'il le demande. Ceci joue un rôle dans l'amélioration de la transparence du SIP et la compréhension et la satisfaction de l'utilisateur.

10.5. Conclusion

La démarche méthodologique, proposée dans ce Chapitre 10, représente un processus de conception d'un SIP dans lequel cohabitent différentes entités. En se basant sur les différentes étapes de cette méthodologie hybride, le concepteur se trouve muni d'un outil l'aidant à décrire les ontologies d'intentions et de contextes nécessaires selon le domaine d'application et à déterminer les espaces de services importants ainsi que ses éléments clés tels que les services sémantiques, les intentions potentielles et les types de contextes nécessaires.

La conception d'un tel espace de services et de ses différentes entités va pouvoir être exploitée par la suite par les mécanismes de découverte et de prédiction de services basés sur ces notions et qui représentent la mise en œuvre de cet espace présenté dans l'architecture de gestionnaire de SIP (*cf.* Chapitre 9).

Chapitre 11. CONCLUSIONS ET PERSPECTIVES

11.1. CONCLUSIONS

Le travail réalisé dans le cadre de cette thèse se positionne dans le domaine de l'*Ingénierie des Services*, de l'*Informatique Pervasive* et des *Systèmes d'Information*. Dans ces domaines, nous nous sommes concentrés sur l'émergence d'une nouvelle génération de SI, les *Systèmes d'information Pervasifs* (SIP). Nous faisons donc face à cette nouvelle génération, qui doit désormais être conçue et mise en œuvre en tant que système sensible au contexte, centrée utilisateur et transparent afin de répondre au mieux aux besoins des utilisateurs.

11.1.1. Rappel de la problématique

Ce travail de thèse part du constat que *l'arrivée de l'Informatique Pervasive, au sein des organisations va impacter directement les Systèmes d'Information (SI)*. La mobilité qu'apportent ces nouvelles technologies étend les SI bien au-delà des frontières physiques de l'organisation. La démocratisation des dispositifs et l'évolution des technologies mobiles bouleversent la manière dont on utilise ces systèmes en apportant avec elles de nouveaux modes d'accès offerts à ces utilisateurs devenus mobiles.

Cette évolution est au cœur de l'apparition des Systèmes d'Information Pervasifs, lesquels représentent une nouvelle génération des SI. Afin de concevoir de tels SI évoluants dans un environnement pervasif, il est nécessaire de bien comprendre les caractéristiques et les exigences auxquelles cette nouvelle génération de SI sera soumise. Il faut prendre en considération : (i) l'*hétérogénéité* caractérisant l'environnement pervasif intégrant les SI ; (ii) la *dynamique* de cet environnement, qui varie en fonction des utilisateurs et de ses éléments ; (iii) les *besoins* des utilisateurs ; et (iv) le caractère *contrôlable et maîtrisable* des SI. Partant de ces faits, les SIP se doivent d'être *sensibles au contexte,* s'adaptant aux changements de l'environnement afin de gérer sa dynamique, et *compréhensibles* à leurs utilisateurs afin d'assurer la *transparence* nécessaire et de gérer l'*hétérogénéité de l'environnement*, sans pour autant perdre complétement le caractère *maîtrisé* et *prédictible* propre aux SI.

A partir de ce constat, nous avons focalisé notre travail sur une problématique *de conception et de réalisation d'un SIP répondant à l'ensemble des besoins de transparence, d'adaptation à l'environnement et d'adaptation à l'utilisateur*. Afin de répondre à cette problématique, il nous semble impérative de répondre aux problèmes suivants :

(1) Comment assurer la transparence nécessaire aux SIP indépendamment de l'hétérogénéité caractérisant les environnements pervasifs ?

Les SIP se caractérisent autant par l'hétérogénéité de l'environnement (des ressources, des infrastructures et des terminaux), que par l'hétérogénéité des services offerts par le SI. Il devient ainsi crucial de faire face à cette hétérogénéité tout en se concentrant sur l'objectif principal de l'utilisateur et non sur la technologie elle-même. Ces systèmes se doivent de gérer l'hétérogénéité des environnements et des services, et ils doivent le faire de la manière la plus *transparente* possible à l'utilisateur. En effet, dans le cadre d'un Système d'Information les utilisateurs doivent se concentrer sur leurs propres activités et non sur la technologie elle-même. La *transparence* devient ainsi un des critères fondateurs d'un SIP puisqu'elle permet de masquer cette hétérogénéité aux utilisateurs. Sans transparence, tout Systèmes d'Information Pervasifs ne sera pas en mesure de remplir avec succès son rôle de SI.

(2) Comment faire face à la dynamique propre aux environnements pervasifs sans perdre le contrôle et la maîtrise d'un SIP ?

Les SIP doivent être conçus pour opérer dans un environnement pervasif qui se caractérise par sa nature hautement *dynamique*. La nature dynamique de ces environnements impose aux SIP une capacité d'adaptation à des conditions d'opération différentes et à des utilisateurs dont les préférences et le comportement sont également variables. Ainsi, afin d'assurer la transparence nécessaire, les SIP doivent être *sensibles au contexte*, permettant la prise en compte de l'environnement pervasif, fournissant ainsi à l'utilisateur le service le plus approprié en fonction de son contexte courant. Cependant, les SIP doivent trouver le juste milieu entre l'adaptation à cette dynamique et le contrôle nécessaires aux SI. Cette équation s'avère assez délicate. Il est nécessaire de *s'adapter aux changements* de l'environnement afin de gérer cette dynamique.

(3) Comment répondre aux besoins des utilisateurs de la manière la plus transparente possible ?

Les SIP doivent être conçus de manière à s'adapter non seulement à leur environnement, mais également à leurs utilisateurs. Ceux-ci ont des besoins auxquels les SIP doivent répondre de la manière la plus adaptée possible, tout en gardant la transparence nécessaire pour que le système disparaisse derrière la satisfaction de ces besoins. En conséquence, pour la bonne réussite d'un SIP, il est important de bien comprendre les besoins des utilisateurs, d'une part, et de répondre d'une façon personnalisée à ces besoins, d'autre part.

11.1.2. Bilan du travail réalisé

Afin de répondre à cette problématique, résumée ci-dessus, nous avons proposé, dans le cadre de cette thèse, une nouvelle vision centrée utilisateur d'un SIP transparent, non intrusif et compréhensible aux besoins de l'utilisateur. Afin de concevoir une telle vision, nous avons choisi de nous baser, dans le cadre de cette thèse, sur les notions d'*intention*, de *contexte* et de *service*, comme des solutions permettant de : (i) gérer l'hétérogénéité et la dynamique de l'environnement pervasif ; (ii) comprendre et assimiler les besoins réels des utilisateurs ; et (iii) garantir le niveau nécessaire de contrôle et de maîtrise dans le cadre d'un SI. Cette vision se base sur l'*orientation service* parce qu'elle permet de répondre au besoin de *gestion de l'hétérogénéité technique* de l'environnement intégrant ce système, grâce à son i*ndépendance par rapport aux aspects technologiques.* De plus, elle est fondée sur la *sensibilité au contexte,* puisqu'elle assure ainsi l'adaptation nécessaire du SIP au contexte de l'utilisateur et à l'environnement, gérant ainsi le dynamique de l'environnement pervasif. Et finalement, cette vision est conçue autour d'une *orientation intentionnelle* laquelle permet de mieux répondre aux besoins de l'utilisateur. Cette notion d'intention est nécessaire afin de mieux comprendre l'utilisateur et de répondre ainsi à ses besoins d'une manière appropriée.

Dans le cadre de cette thèse, afin d'explorer la relation entre l'intention, le contexte et le service, nous avons poussé nos travaux plus loin en exprimant cette relation dans les descriptions de services et en l'exploitant dans les processus de découverte et de prédiction de services. Nos contributions, pour mettre en place cette vision intentionnelle et contextuelle des SIP, se déclinent sur quatre dimensions principales (conceptuelle, fonctionnelle, système et support). Elles se résument autour des quatre points suivants :

(1) ***Un cadre conceptuel de SIP (dimension conceptuelle)***

Etant donné qu'aucun cadre conceptuel de SIP n'a été proposé précédemment dans le domaine des Systèmes d'Information et dans le domaine de l'Informatique Pervasive, nous avons proposé, dans cette thèse, la notion d'*espace de services,* laquelle se veut un cadre conceptuel pour la définition des SIP. Aujourd'hui, les concepteurs des SIP se trouvent démunis face à une notion relativement nouvelle qui est difficile à conceptualiser, avec peu de formalismes disponibles. Autant dire qu'ils n'ont rien à leur disposition pour les aider à concevoir et mettre en place de tels systèmes. Cette notion permet ainsi d'identifier les différents éléments d'un SIP représentant les *services* (entités actives) offerts par celui-ci, ainsi qu'un ensemble de *capteurs de contexte* (entités passives) qui renseignent l'utilisateur et le système sur l'environnement qui les entoure. Nous proposons, par cette notion d'espace de services dynamique et perméable, un outil conceptuel permettant d'aider le concepteur à identifier les descriptions des services qui seront offert à l'utilisateur par le SIP.

Par la proposition d'un tel cadre conceptuel indépendant de la technique, nous proposons un outil conceptuel permettant de gérer l'hétérogénéité des environnements pervasifs et des SIP déployés sur ces environnements, ce qui offre aux concepteurs des SIP les moyens

d'analyser de manière abstraite l'interaction entre un utilisateur et le système facilitant ainsi l'identification des différents éléments du SIP.

(2) ***Des mécanismes de découverte et de prédiction de services (dimension fonctionnelle)***

Afin de gérer l'interaction de l'utilisateur avec son SIP, dans le cadre de l'espace de services, nous proposons des mécanismes de découverte et de prédiction de services guidés par l'intention et le contexte. Ces mécanismes permettent de satisfaire les besoins de l'utilisateur en lui offrant le service le plus approprié, selon son intention et son contexte, assurant ainsi la dynamique et la pro-activité des SIP. D'une part, nous avons proposé un mécanisme de découverte de services dynamique, que nous estimons essentiel dans le cadre de SIP, parce qu'il permet de répondre aux besoins immédiats de l'utilisateur d'une manière personnalisée. Dans ce mécanisme, nous nous appuyons sur la notion d'*intention* afin de mieux comprendre le besoin réel derrière la demande de l'utilisateur. Nous nous basons également sur la notion de *contexte* afin de découvrir le service capable de répondre au mieux au contexte courant de l'utilisateur, assurant ainsi une meilleure adaptation à l'environnement. D'autre part, nous avons proposé également un mécanisme de prédiction de services, introduisant ainsi des techniques de recommandation afin d'augmenter le caractère dynamique et proactif des SIP. Ce mécanisme permet d'anticiper, à partir de son historique, les besoins futurs de l'utilisateur et de lui suggérer, d'une manière transparente, le service le plus approprié qui pourra, par la suite, l'intéresser. Ce processus gère les traces de l'utilisateur afin de pouvoir modéliser dynamiquement ses habitudes, permettant ainsi de lui suggérer les services les mieux adaptés qui peuvent l'intéresser. Ceci permet d'offrir une meilleure pro-activité au SIP, contribuant ainsi à l'amélioration de la transparence nécessaire des SIP.

Notre contribution se résume à une découverte transparente des services, dans laquelle l'utilisateur se concentre uniquement sur ces besoins et non sur les fonctionnalités techniques, contrairement aux mécanismes de découverte de services traditionnels. De plus, l'originalité de notre processus de prédiction de services, comparé aux autres approches, et que nous tirons profit des traces de l'utilisateur, lesquelles représentent ses situations (<intention, contexte, service>) à des instants précis, afin modéliser ses habitudes. Dans ce travail, nous partons de l'hypothèse que des situations similaires peuvent être détectées, surtout dans le cadre d'un Systèmes d'Information Pervasifs.

(3) ***Une architecture de gestionnaire de SIP (dimension système)***

Afin de mettre en place cette vision intentionnelle et contextuelle des SIP, nous avons proposé une *architecture de gestionnaire de SIP* conforme au cadre conceptuel. Cette architecture intègre nos différentes propositions pour la construction d'un SIP transparent et centré utilisateur. Elle intègre le gestionnaire de contexte, un répertoire de services décrits sémantiquement selon notre extension ***OWL-SIC*** (*OWL-S Intentional & Contextual*)**,** cachant ainsi la complexité technique de l'environnement en décrivant sémantiquement les services selon le contexte dans lequel ils s'adaptent au mieux et les intentions qu'ils permettent de

satisfaire. De plus, cette architecture intègre la mise en œuvre du ***processus de découverte de services guidé par l'intention et le contexte*** et celle du ***processus de prédiction de services intentionnel et contextuel*** permettant de répondre au mieux aux besoins immédiats et futurs de l'utilisateur, d'une manière non intrusive.

Ainsi cette *architecture de gestionnaire de SIP* permet de mettre en œuvre l'objectif principal de notre thèse, à savoir la vision intentionnelle et contextuelle des SIP en mettant en place ses différentes dimensions conformément à l'*espace de services*.

(4) ***Une démarche méthodologique (dimension support)***

Afin de guider le concepteur dans la conception et dans l'identification des différents éléments d'un SIP, nous avons défini *une démarche méthodologique*. Cette démarche va aider le concepteur à définir ses espaces de services, le modèle de contexte nécessaire, ainsi que la description sémantique de ces services. Cette démarche est constituée d'un ensemble d'étapes à suivre commençant par la spécification des différents espaces de services envisageables jusqu'à la description des services proposés au sein de l'architecture de gestionnaire de SIP. Ces étapes vont permettre au concepteur de spécifier les fonctionnalités attendues de leur SIP, ainsi que les informations contextuelles qui seront capturées par celui-ci.

Afin d'illustrer le déroulement de cette méthodologie, et ainsi l'application des différents éléments proposés dans cette thèse, un cas d'étude a été proposé.

11.2. PERSPECTIVES

Ce travail de thèse soulève de nouvelles questions de recherche. Il ouvre la voie à de nouvelles perspectives que nous considérons intéressantes. Nous soulignons dans la suite, certaines de ces perspectives qui vont contribuer à l'évolution des propositions que nous avons réalisées dans le cadre de cette thèse. Ces perspectives s'organisent comme suit :

(1) Expérimentation en entreprise des concepts proposés pour le développement de SIP et la finalisation de l'outillage supportant ces concepts

Nous envisageons, à long terme, d'appliquer l'approche proposée en entreprise pour mettre en œuvre le côté pervasif du SI de manière transparente et centrée utilisateur. Ceci nécessite une phase d'expérimentation des différents concepts, que nous avons proposés dans le cadre de cette thèse, pour la conception et la construction de SIP. De plus, une amélioration des outils supportant cette approche doit être réalisée pour être adoptée en entreprise.

La concrétisation de notre proposition dans l'entreprise soulève certains points d'optimisations des implémentations et des mécanismes que nous avons développés, qui ne sont pas dans le cadre de cette thèse. Ces points peuvent se résumer comme suit :

(1.1) L'optimisation de l'implémentation des mécanismes de découvertes et de prédiction de services

Dans le cadre de cette thèse, nous avons proposé une première version d'implémentation de nos processus de découverte et de prédiction de services, sur laquelle nous avons mené nos expérimentations pour valider la faisabilité de notre proposition. Cette première implémentation nous a donné plusieurs pistes d'optimisation du code de développement. Ainsi, dû au fait que l'algorithme de découverte de services, par exemple, fonctionne service par service, l'usage des threads en JAVA, lesquels permettent de comparer en parallèle plusieurs services, pourrait ainsi améliorer le temps d'exécution de cet algorithme. Toutefois, l'utilisation des threads pose la question de l'accès synchrone aux modèles et aux ontologies gardés en mémoire. Le problème qui se pose ici est de gérer l'accès multiple à ces modèles. Ceci ne représente pas une tâche facile, mais une tâche qui mérite de la réflexion. Nous pouvons, par exemple, utiliser différentes machines pour répartir le traitement des services.

(1.2) La prise en compte de la maintenance des clusters

Le processus de *clustering* que nous proposons dans le cadre de cette thèse, se charge de grouper des observations similaires dans des mêmes *clusters*. Ces *clusters* doivent être maintenus et mis à jour dynamiquement. Cette maintenance implique un certain nombre d'activités pas toujours triviales, à savoir l'élimination des *clusters* non pertinents, la suppression des observations des *clusters* lors de leur suppression de la base des traces, la pénalisation des *clusters* qui n'ont pas été mis à jour, la récompense des *clusters* qui se mettent à jour fréquemment, etc. Nous envisageons, à court terme, de développer un processus de maintenance qui se charge de mettre à jour les *clusters* dynamiquement afin d'assurer leur pertinence.

(1.3) La gestion de l'aspect dynamique de l'espace de services

Dans le cadre de cette thèse, la disponibilité des services est représentée par la présence du service dans le répertoire de services sémantique. Toutefois, l'idéal serait de gérer dynamiquement cette disponibilité par la gestion de contexte tel que défini clairement dans l'espace de services. Le problème que nous soulevons ici est qu'il faut observer le service, gérer son contexte d'exécution, etc. Ce travail nécessite une plateforme de monitoring de contexte qui n'est pas dans le cadre de cette thèse, mais qui représente une de nos perspectives à long terme.

Ensuite, afin d'étendre notre proposition à un domaine plus ouvert et un système plus dynamique, nous envisageons d'exploiter la dimension intentionnelle pour améliorer l'agilité des services et être capable de faire de la composition dynamique de services lors de la découverte et prédiction de services. Plus précisément, l'objectif est de reconstruire dynamiquement la composition intentionnelle de services et la variabilité intentionnelle, lors de la découverte et la prédiction de services.

(2) La composition dynamique de services selon l'intention et le contexte

La description de services que nous proposons ouvre de nouvelles perspectives à travers la composition intentionnelle décrite par les processus intentionnels, qui n'ont pas été abordées dans le cadre de cette thèse. Le fait d'avoir la composition intentionnelle dans la description de services pourrait nous permettre de composer de manière dynamique le service à la fois selon le contexte et selon l'intention.

Une première piste pourrait découler de la prédiction de services laquelle permet de trouver des compositions simples de services à travers les traces. En réalité nous avons une forme de composition simple dans les traces de l'utilisateur. Le processus de prédiction de services (y compris le processus d'apprentissage) pourrait nous permettre de construire des compositions. Par contre, ceci engendre, techniquement, un problème d'interopérabilité entre les services. En effet, deux intentions compatibles ne rendent pas forcément deux services compatibles techniquement. Cette piste est à étudier à moyen terme afin de gérer dynamiquement la composition des services agrégats selon leurs intention et contexte.

Cette composition dynamique de services va améliorer la découverte et la prédiction de services et permet de s'ouvrir à un domaine d'application plus ouvert.

(3) La gestion de la variabilité intentionnelle

Dans nos descriptions de services, selon notre extension OWL-SIC, nous représentons la variabilité intentionnelle. Nous partons de l'hypothèse que nous travaillons dans le cadre d'un SI, lequel représente un cadre fermé avec des ontologies de domaine bien établies. Toutefois, ça aurait pu être intéressant d'étendre cette démarche sur d'autres systèmes plus ouverts (les applications de manière générale, où nous n'avons pas d'ontologies bien définies). Dans ce cadre ouvert, l'utilisateur ne va pas exprimer de la même manière son besoin. Ceci implique la nécessité d'avoir différents moyens pour comparer ces intentions (et pas seulement aider l'utilisateur à reformuler son besoin, comme dans les travaux de Aljoumaa (Aljoumaa, 2011)). Notre démarche doit aller au-delà des ontologies de domaine et gérer tout l'aspect linguistique, par exemple le traitement de texte, afin de gérer dynamiquement la variabilité dans l'expression de l'intention.

BIBLIOGRAPHIE

Abbar, S., Bouzeghoub, M., et Lopez, S. (2009). Context-Aware Recommender Systems: A Service-Oriented Approach. In 3rd International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases (PersDB), Lyon, France.

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A., et Verma, K. (2005). Web Service Semantics - WSDL-S. W3C Member Submission. Disponible sur http://www.w3.org/Submission/WSDL-S/

Ait Ali Slimane, A. (2012). Méthode de sélection intentionnelle des services, en fonction des exigences non-fonctionnelles des agents métier. Thèse de doctorat, Université Panthéon-Sorbonne Paris I.

Aljoumaa, K. (2011). Modélisation intentionnelle et annotation sémantique pour la réutilisation de services métiers. Thèse de doctorat, Université Panthéon-Sorbonne - Paris I.

Aljoumaa, K., Assar, S., et Souveyet, C. (2011). Reformulating User's Queries for Intentional Services Discovery Using an Ontology-Based Approach. In 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–4.

Allen, P., et Frost, S. (1998). Component-based development systems : applying the SELECT perspective. Cambridge : Cambridge University Press and Sigs Books, 496 p.

Almeida, V.A.F., et Menasce, D. (2002). Capacity planning an essential tool for managing Web services. IT Professional. 4(4), pp. 33–38.

Alonso, G., Casati, F., Kuno, H., et Machiraju, V. (2004). Web Services : Concepts, Architectures and Applications. Springer, 354 p.

Alter, S. (1992). Information systems : a management perspective. Addison-Wesley Pub. Co., 848 p.

Antón, A.I., McCracken, W.M., et Potts, C. (1994). Goal Decomposition and Scenario Analysis in Business Process Reengineering. In Proceedings of the 6th International Conference Advanced Information Systems Engineering (CAiSE), Utrecht, The Netherlands, pp. 94–104.

Austin, D., Barbir, A., Ferris, C., et Garg, S. (2002). Web Services Architecture Requirements. W3C Working Group Note. Disponible sur http://www.w3.org/TR/wsa-reqs/

Baldauf, M., Dustdar, S., et Rosenberg, F. (2007). A survey on context-aware systems. International Journal of Ad Hoc Ubiquitous Comput, *2(4)*, pp. 263–277.

Banâtre, M., Bryce, C., Couderc, P., et Weis, F. (2007). Informatique diffuse : des concepts à la réalité. Hermes Science Publications, 204 p.

Bauer, J. (2003). Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic. Thèse de doctorat, Technische Universitat Berlin Fakultat IV - Elektrotechnik und Informatik Institut fur Computergestuutzte Informations systeme.

Beckett, D. (2004). RDF/XML Syntax Specification. W3C Recommendation. Disponible sur http://www.w3.org/TR/REC-rdf-syntax/

Bell, G., et Dourish, P. (2007). Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. Journal Personal and Ubiquitous Computing, *11(2)*, pp. 133–143.

Bellwood, T. (2002). UDDI Version 2.04 API Specification. UDDI Committee Specification. Disponible sur http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm

Berners-Lee, T., Hendler, J., et Lassila, O. (2001). The Semantic Web. In Scientific Am., *284(5)*, pp. 34-43.

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., et Riboni, D. (2010). A survey of context modelling and reasoning techniques. In Pervasive Mobile Computing, *6(2)*, pp. 161–180.

Birnbaum, J. (1997). Pervasive information systems. Communications of the ACM, *40(2)*, pp. 40–41.

Bolchini, C., Curino, C.A., Orsi, G., Quintarelli, E., Rossato, R., Schreiber, F.A., et Tanca, L. (2009). And what can context do for data? Communications of the ACM, *52(11)*, pp.136–140.

Bondi, A.B. (2000). Characteristics of scalability and their impact on performance. In Proceedings of the 2nd International Workshop on Software and Performance, (New York, NY, USA: ACM), pp. 195–203.

Booth, D., Haas, H., Ferris, C., McCabe, F., Newcomer, E., Orchard, D., et Champion, M. (2004). Web Services Architecture. W3C Working Group Note. Disponible sur http://www.w3.org/TR/ws-arch/

Bouzy, B., et Cazenave, T. (1997). Using the Object Oriented Paradigm to Model Context in Computer Go. In Proceedings of Context.

Boytsov, A., et Zaslavsky, A. (2011). Context Prediction in Pervasive Computing Systems : Achievements and Challenges. In Supporting Real Time Decision-Making, F. Burstein, P. Brézillon, and A. Zaslavsky, eds. (Springer US), pp. 35–63.

Boytsov, A., Zaslavsky, A., et Synnes, K. (2009). Extending Context Spaces Theory by Predicting Run-Time Context. In Proceedings of the 9th International Conference on Smart Spaces and Next Generation Wired/Wireless Networking (NEW2AN) and Second Conference on Smart Spaces (ruSMART), 5764, (Berlin, Heidelberg: Springer-Verlag), pp. 8–21.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., et Mylopoulos, J. (2004). Tropos: An Agent-Oriented Software Development Methodology. In Journal Autonomous Agents Multi-Agent Systems, *8(3)*, pp. 203–236.

Brézillon, P. (2002). Expliciter le contexte dans les objets communicants. In Objets Communicants, C. Kintzig, G. Poulain, G. Privat, et P.N. Favennec, eds. (Hermes Science Publications), pp. 295–303.

Brézillon, P. (2005). Task-realization models in contextual graphs. In Modeling et Using Context. In 5th International et Interdisciplinary Conference CONTEXT, 3554 of Lecture Notes in Computer Science, (Springer Verlag), pp. 55–68.

Brickley, D., et Guha, R.V. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation. Disponible sur http://www.w3.org/TR/rdf-schema/

Brickley, D., et Miller, L. (2005). FOAF Vocabulary Specification. Namespace Document. Disponible sur http://xmlns.com/foaf/spec/.

Bronsted, J., Hansen, K.M., et Ingstrup, M. (2010). Service Composition Issues in Pervasive Computing. IEEE Pervasive Computing, *9(1)*, pp. 62–70.

Brown, P.J., Bovey, J.D., et Chen, X. (1997). Context-aware applications : from the laboratory to the marketplace. IEEE Personal Communications. *4(5)*,pp. 58–64.

De Bruijn, J., Lausen, H., Polleres, A., et Fensel, D. (2006). The Web service modeling language WSML: an overview. In Proceedings of the 3rd European Conference on The Semantic Web: Research and Applications, (Berlin, Heidelberg: Springer-Verlag), pp. 590–604.

Buchholz, S., Hamann, T., et Hübsch, G. (2004). Comprehensive Structured Context Profiles (CSCP): Design and Experiences. In Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW), (Washington, DC, USA: IEEE Computer Society), p. 43.

Burges, C.J.C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. In Journal Data Mining and Knowledge Discovery, *2(2)*, pp. 121–167.

Bussler, C., Cimpian, E., Fensel, D., Gomez, J.M., Haller, A., et Haselwanter, T. (2005). Web Service Execution Environment (WSMX). W3C Member Submission. Disponible sur http://www.w3.org/Submission/WSMX/

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Souchon, N., Bouillon, L., Florins, M., et Vanderdonckt, J. (2002). Plasticity of User Interfaces : A Revised Reference Framework. In Proceedings of the First International Workshop on Task Models and Diagrams for User Interface Design, (INFOREC Publishing House Bucharest), pp. 127–134.

Carrillo-Ramos, A., Kirsch-Pinheiro, M., Villanova-Oliver, M., Gensel, J., et Berbers, Y. (2009). Collaborating agents for adaptation to mobile users. In Collaborative and Social Information Retrieval and Access: Techniques for Improved User Modeling, M. Chevalier, and C. Soule-Dupuy, eds. (IGI global), pp. 250–276.

Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., et Wilkinson, K. (2004). Jena: implementing the semantic Web recommendations. In Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, (New York, USA: ACM), pp. 74–83.

Chaari, T. (2007). Adaptation d'applications pervasives dans des environnements multi-contextes. Thèse de doctorat, Institut national des sciences appliquées de Lyon

Chaari, T., Laforest, F., et Celentano, A. (2004). Design of context-aware applications based on Web services, Rapport de recherche RR-LIRIS, Lyon.

Chaari, T., Laforest, F., et Flory, A. (2005). Adaptation des applications au contexte en utilisant les services Web. In Proceedings of the 2nd French-speaking Conference on Mobility and Ubiquity Computing (Ubimob), (New York, NY, USA: ACM), pp. 111–118.

Chaari, S., Badr, Y., et Biennier, F. (2008a). Enhancing Web service selection by QoS-based ontology and WS-policy. In Proceedings of the 2008 ACM Symposium on Applied Computing, (New York, NY, USA: ACM), pp. 2426–2431.

Chaari, T., Laforest, F., et Celentano, A. (2008b). Adaptation in context-aware pervasive information systems: the SECAS project. In International Journal Pervasive Computing and Communications, *3(4)*, pp. 400–425.

Chen, H. (2004). An Intelligent Broker Architecture for Pervasive Context-Aware Systems. Thèse de doctorat, University of Maryland, Baltimore County.

Chen, G., et Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research. TR2000-381. Dept. of Computer Science, (Hanover, NH, USA: Dartmouth College).

Chen, H., Finin, T., et Joshi, A. (2003). An Ontology for Context-Aware Pervasive Computing Environments. In Journal the knowledge engineering, *18(3)*, pp197–207.

Chen, H., Perich, F., Finin, T., et Joshi, A. (2004a). SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In International Conference on Mobile and Ubiquitous Systems : Networking and Services, pp. 258–267.

Chen, H., Finin, T., et Joshi, A. (2004b). Semantic Web in the context broker architecture. In Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications (PerCom), pp. 277–286.

Cheng, G. (2010). Future living Environment. In Pervasive Adaptation: The Next Generation Pervasive Computing Research Agenda, A. Ferscha, ed. (Inst. for Pervasive Computing, Johannes Kepler Univ.), p. 16.

Chinnici, R., Moreau, J.-J., Ryman, A., et Weerawarana, S. (2007). Web Services Description Language (WSDL) Version 2.0. W3C Recommendation. Disponible sur http://www.w3.org/TR/wsdl20/

Christensen, E., Curbera, F., Meredith, G., et Weerawarana, S. (2001). Web Services Description Language (WSDL) 1.1. W3C Note. Disponible sur http://www.w3.org/TR/wsdl

Conan, D., Rouvoy, R., et Seinturier, L. (2007). Scalable Processing of Context Information with COSMOS. In Proceedings of the 7th IFIP WG 6.1 International Conference on Distributed applications and interoperable systems (DAIS), pp. 210–224.

Corby, O., Faron-Zucker, C., et Mirbel, I. (2009). Démarche sémantique de recherche d'information sur le Web. In 20es Journées Francophones d'Ingénierie des Connaissances (IC) " Connaissance et communautés en ligne " F.L. Gandon, ed. (Tunisie), pp. 289–300.

Coronato, A., Esposito, M., et Pietro, G.D. (2009). A multimodal semantic location service for intelligent environments: an application for Smart Hospitals. In Journal Personal and Ubiquitous Computing. *13(7)*, pp. 527–538.

Curbera, F., Nagy, W.A., et Weerawarana, S. (2001). Web Services: Why and How. In OOPSLA Workshop on Object-Oriented Web Services. ACM.

Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., et Weerawarana, S. (2002). Unraveling the Web services: an introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing. *6(2)*, pp. 86–93.

Dardenne, A., Lamsweerde, A. van, et Fickas, S. (1993). Goal-directed Requirements Acquisition. In Journal Science of Computer programming, *20(1-2)*, pp. 3–50.

Daszykowski, M., Walczak, B., et Massart, D.L. (2002). On the Optimal Partitioning of Data with K-Means, Growing K-Means, Neural Gas, and Growing Neural Gas. Journal of Chemical Information and Modelling, *42(6)*, pp. 1378–1389.

Deneckere, R., et Kornyshova, E. (2010). La variabilité due à la sensibilité au contexte dans les processus téléologiques. In Actes du 18° Congrés INFORSID, pp. 161–176.

Dey, A.K. (2000). Providing architectural support for building context-aware applications. Georgia Institute of Technology. Thèse de doctorat, Georgia Institute of Technology.

Dey, A.K. (2001). Understanding and Using Context. Journal Personal Ubiquitous Computing, *5(1)*, pp. 4–7.

Dey, A.K. (2011). Intelligibility in ubiquitous computing systems. In Pervasive Adaptation: The Next Generation Pervasive Computing Research Agenda, A. Ferscha, ed. (Inst. for Pervasive Computing, Johannes Kepler Univ.), pp. 68–69.

Dey, A.K., et Abowd, G.D. (2000). Towards a Better Understanding of Context and Context-Awareness. In Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing (HUC), pp. 1–12.

Dey, A.K., Abowd, G.D., et Wood, A. (1998). CyberDesk : a framework for providing self-integrating context-aware services. In Proceeding of the 3rd international conference on Intelligent user interfaces (IUI), pp. 47-54

Dik, S.C. (1989). The theory of functional grammar (Foris Publications). Dordrecht, Holland ; Providence, RI, U.S.A. : Foris Publications, p. 497

Dourish, P. (2004). What we talk about when we talk about context. Journal Personal and Ubiquitous Computing, *8(1)*, pp. 19–30.

Farrell, J., et Lausen, H. (2007). Semantic Annotations for WSDL and XML Schema (SAWSDL).W3C Recommendation. Disponible sur http://www.w3.org/TR/sawsdl/

Feller, W. (1968). An introduction to probability theory and its applications. (New York ; London ; Sydney: J. Wiley & sons), 509 p.

Fensel, D., Bussler, C., Ding, Y., et Omelayenko, B. (2002). The Web Service Modeling Framework WSMF. Electronic Commerce Research and Applications, *1(2)*, pp. 113–137.

Fensel, D., Facca, F.M., Simperl, E., et Toma, I. (2011). Semantic Web Services (Berlin : Springer), 350 p.

Fillmore, C.J. (1968). The case for case. In Bach et Harms, eds. (New York: Holt, Rinehart, and Winston), *Universals in Linguistic Theory,* 88 p.

Fremantle, P., Weerawarana, S., et Khalaf, R. (2002). Enterprise services. Communication of the ACM, *45(10)*, pp. 77–82.

Friedman, N., Geiger, D., et Goldszmidt, M. (1997). Bayesian Network Classifiers. Journal Machine learning - Special issue on learning with probabilistic representations, *29(2-3)*, pp. 131–163.

Fries, B., Khalid, M., et Klusch, M. (2005). OWL-S Service Retrieval Test Collection. Disponible sur http://projects.semwebcentral.org/projects/owls-tc/

Frioui, S. (2012). Découverte intentionnelle et contextuelle de services. université Paris 1 Panthéon-Sorbonne. Mémoire de Master, Université Paris1 Panthéon-Sorbonne.

Gamma, E., Helm, R., Johnson, R., et Vlissides, J. (1994). Design patterns : elements of reusable object-oriented software (Boston: Addison-Wesley), 395 p.

Geihs, K., Reichle, R., Wagner, M., et Khan, M.U. (2009). Modeling of Context-Aware Self-Adaptive Applications in Ubiquitous and Service-Oriented Environments. In Software Engineering for Self-Adaptive Systems, B.H.C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, eds. (Springer Berlin Heidelberg), pp. 146–163.

Gensel, J., Villanova-Oliver, M., et Kirsch-Pinheiro, M. (2008). Modèles de contexte pour l'adaptation à l'utilisateur dans des Systèmes d'Information Web collaboratifs. In 8èmes Journées Francophones d'Extraction et Gestion Des Connaissances (EGC'08), Atelier Sur La Modélisation Utilisateur et Personalisation d'Interfaces Web, pp. 5–15.

Gomez, J.M., Rico, M., García-Sanchez, F., Toma, I., et Han, S.-K. (2006). GODO : Goal Oriented Discovery for Semantic Web Services. In 5th international Semantic Web Conference.

Gopalratnam, K., et Cook, D.J. (2003). Active Lezi : An incremental parsing algorithm for sequential prediction. In Sixteenth International Florida Artificial Intelligence Research Society Conference, pp. 38–42.

Granell, C., Díaz, L., et Gould, M. (2010). Service-oriented applications for environmental models: Reusable geospatial services. Journal Environmental Modelling and Software, *25(2)*, pp. 182–198.

Greenberg, S. (2001). Context as a dynamic construct. Journal Human Computer Interaction, *16(2)*, pp. 257–268.

Greenfield, A. (2006). Everyware : the dawning age of ubiquitous computing. (Berkeley, CA: New Riders), 272 p.

Grenon, P. (2009). Defining extensions to WSMO for capturing contextual information. Disponible sur http://www.soa4all.eu/pdocs/deliverables/D3.4.7+DEFINING+EXTENSIONS+WSMO+CAPTURING+CONTEXT+INFO.PDF

Grimm, S., Keller, U., Lausen, H., et Nagypál, G. (2007). A Reasoning Framework for Rule-Based WSML. In The Semantic Web: Research and Applications, E. Franconi, M. Kifer, and W. May, eds. (Springer Berlin Heidelberg), pp. 114–128.

Groot, B. de, et Welie, M. van (2002). Leveraging the Context of Use in Mobile Service Design. In Human Computer Interaction with Mobile Devices, F. Paternò, eds. (Springer Berlin Heidelberg), pp. 334–338.

Gruber, T.R. (1993). A translation approach to portable ontology specifications. Knowledge Acquisition. *5(2)*, pp. 199–220.

Gu, T., Wang, X.H., Pung, H.K., et Zhang, D.Q. (2004). An Ontology-based Context Model in Intelligent Environments. In Proceeding of communication networks and distributed systems modeling and simulation conference, pp. 270–275.

Haarslev, V., et Möller, R. (2003). Racer: A Core Inference Engine for the Semantic Web. Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), pp. 27–36.

Hagras, H. (2011). Intelligent pervasive adaptation in shared spaces. In Pervasive Adaptation: The Next Generation Pervasive Computing Research Agenda, A. Ferscha, eds. (Inst. for Pervasive Computing, Johannes Kepler Univ.), pp. 16–17.

Hall, R., Pauls, K., McCulloch, S., et Savage, D. (2011). OSGi in Action : Creating Modular Applications in Java (Manning Publications), 375 p.

Halpin, T. (2001). Information modeling et relational databases from conceptual analysis to logical design (San Francisco: Morgan Kaufman Publishers), 761 p.

Hamker, F.H. (2001). Life-long learning cell structures--continuously learning without catastrophic interference. Journal Neural Networks, *14(4-5)*, pp. 551–573.

Han, L., Jyri, S., Ma, J., et Yu, K. (2008). Research on Context-Aware Mobile Computing. In 22nd International Conference on Advanced Information Networking and Applications - Workshops (AINAW), pp. 24–30.

Harold, W., et Lochart, J. (1994). OSF DCE: guide to developing distributed applications (New York, NY, USA: McGraw-Hill, Inc.), 561 p.

Held, A., Buchholz, S., Schill, A., et Schill, E. (2002). Modeling of Context Information for Pervasive Computing Applications. In Proceedings of the First International Conference on Pervasive Computing, pp. 167-180

Henderson, J.C., et Venkatraman, N. (1993). Strategic alignment: leveraging information technology for transforming organizations. IBM System Journal, *32(1)*, pp. 4–16.

Henricksen, K., et Indulska, J. (2004). Modelling and using imperfect context information. In Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 33–37.

Henricksen, K., et Indulska, J. (2006). Developing context-aware pervasive computing applications: Models and approach. Journal Pervasive Mobile Computing, *2(1)*, 37–64.

Henricksen, K., Indulska, J., et Rakotonirainy, A. (2002). Modeling Context Information in Pervasive Computing Systems. In Pervasive Computing, F. Mattern, and M. Naghshineh, eds. (Springer Berlin Heidelberg), pp. 167–180.

Hobbs, J.R. (2002). Towards an ontology for time for the semantic Web. In Proceeding Workshop on Annotation Standards for Temporal Information in Natural Language (LREC), Las Palmas, Spain.

Horrocks, I., Patel-Schneider, P.F., et Harmelen, F.V. (2003). From SHIQ and RDF to OWL: The Making of a Web Ontology Language. Journal Web Semantics : Science, Services and Agents on the World Wide Web, *1(1),* pp. 7–26

Hsu, W.H., Gettings,, N.D., Lease, V.E., Pan, Y., et Wilkins, D.C. (1998). Heterogeneous Time Series Learning for Crisis Monitoring. In Predicting the Future : Ai Approaches to Time-series Problems. Workshop Held in Conjunction with the Fifteenth National Conference on Artificial Intelligence, pp. 34–41.

Huhns, M.N., et Singh, M.P. (2005). Service-oriented computing : key concepts and principles. IEEE Internet Computing, *9(1)*, pp. 75–81.

Hull, R., Neaves, P., et Bedford-Roberts, J. (1997). Towards situated computing. In First International Symposium on Wearable Computers (ISWC), pp. 146–153.

Indulska, J., et Sutton, P. (2003). Location management in pervasive systems. In Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers, *21*, (Darlinghurst, Australia, Australia : Australian Computer Society, Inc.), pp. 143–151.

Issarny, V., Caporuscio, M., et Georgantas, N. (2007). A Perspective on the Future of Middleware-based Software Engineering. In Future of Software Engineerin (FOSE), pp. 244–258.

IST-MUSIC (2010). IST-MUSIC project. http://ist-music.berlios.de/site/

Jackson, M. (1995). Software requirements specifications : a lexicon of practice, principles and prejudices (New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.), 228 p.

Jaege, M.C. (2001). The TUB OWL-S Matcher (The OWLSM). http://owlsm.projects.semwebcentral.org/

Jensen, F.V. (2001). Bayesian networks and decision graphs (Springer), 463 p.

Jones, Q., et Grandhi, S.A. (2005). P3 systems : putting the place back into social networks. IEEE Internet Computing, *9(5)*, pp. 38–46.

Kaabi, R.S. (2007). Une Approche Méthodologique pour la Modélisation Intentionnelle des Services et leur Opérationnalisation. Thèse de doctorat. Université Panthéon-Sorbonne - Paris I.

Kaabi, R.S., et Souveyet, C. (2007). Capturing Intentional services with Business Process Maps. In 1rst IEEE International Conference on Research Challenges in Information Science (RCIS), pp. *309-318*

Keen, M., Acharya, A., Bishop, S., Hopkins, A., Milinski, S., Nott, C., Robinson, R., Adams, J., and Verschueren, P. (2004). IBM Redbooks | Patterns : Implementing an SOA using an Enterprise Service Bus.

Keith, M., Goul, M., Demirkan, H., Nichols, J., et Mitchell, M.C. (2006). Contextualizing Knowledge Management Readiness to Support Change Management Strategies. In Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS), p. 152a.

Keller, U., Lara, R., Polleres, A., Toma, I., Kifer, M., et Fensel, D. (2004). WSMO Web Service Discovery. WSML Working Draft. Disponible sur http://www.wsmo.org/2004/d5/d5.1/v0.1/20041112/

Kerrigan, M. (2005). WSML Editor. WSMX Working Draft. Disponible sur http://www.wsmo.org/TR/d9/d9.2/v0.1/20050321/

Khan, M.U. (2010). Unanticipated Dynamic Adaptation of Mobile Applications. Thèse de doctorat, kassel university.

Kirsch-Pinheiro, M., Gensel, J., et Martin, H. (2004). Representing Context for an Adaptative Awareness Mechanism. In Proceeding of the International Workshop on Groupware (CRIWG), LNCS 3198, pp. 339–348.

Kirsch-Pinheiro, M. (2006). Adaptation Contextuelle et Personnalisée de l'Information de Conscience de Groupe au sein des Systèmes d'Information Coopératifs. Thèse de doctorat, Université Joseph-Fourier - Grenoble I.

Kirsch-Pinheiro, M., Vanrompay, Y., et Berbers, Y. (2008). Context-aware service selection using graph matching. In 2nd Non Functional Properties and Service Level Agreements in Service Oriented Computing Workshop (NFPSLA-SOC'08) at ECOWS, (Dublin, Ireland).

Kirsch-Pinheiro, M., Le Grand, B., Souveyet, C., et Najar, S. (2013). Espace de Services : Vers une formalisation des Systèmes d'Information Pervasifs. In Actes du XXXIème Congrès INFORSID, (Paris), pp. 215–223.

Kiss, C. (2010). Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0. W3C Working Draft. Disponible sur http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430/

Kitsios, F., Papadopoulos, T., et Angelopoulos, S. (2010). A Roadmap to the Introduction of Pervasive Information Systems in Healthcare. Int Journal of Advanced Pervasive Ubiquitous Computing, *2(3)*, pp. 21–32.

Klein, M., et König-Ries, B. (2004). Coupled Signature and Specification Matching for Automatic Service Binding. In Web Services, L.-J. (LJ) Zhang, and M. Jeckle, eds. (Springer Berlin Heidelberg), pp. 183–197.

Klusch, M., et Kaufer, F. (2008). WSMO-MX: A Hybrid Semantic Web Service Matchmaker. In 6th World Congress on Services

Klusch, M., Fries, B., et Sycara, K. (2006). Automated semantic Web service discovery with OWLS-MX. In Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, (New York, NY, USA: ACM), pp. 915–922.

Klusch, M., Kapahnke, P., et Kaufer, F. (2008). Evaluation of WSML Service Retrieval with WSMO-MX. In IEEE International Conference on Web Services (ICWS), pp. 401–408.

Klusch, M., Kapahnke, P., et Zinnikus, I. (2009). Hybrid Adaptive Web Service Selection with SAWSDL-MX and WSDL-Analyzer. In Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications, (Berlin, Heidelberg: Springer-Verlag), pp. 550–564.

Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M.H., et Tran, L. (2004). Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation. Disponible sur http://www.w3.org/TR/CCPP-struct-vocab/

Kohonen, T. (1995). Self-organizing maps (Berlin: Springer), 532 p.

Kourouthanassis, P.E., et Giaglis, G.M. (2006). A Design Theory for Pervasive Information Systems. In 3rd International Workshop on Ubiquitous Computing during *ICEI*, pp. 62–70.

Kourouthanassis, P.E., et Giaglis, G.M. (2007). Toward Pervasiveness: Four Eras of Information Systems Development,. In Pervasive Information Systems, P.E. Kourouthanassis, and G.M. Giaglis, eds. (M.E. Sharpe), pp. 3–15.

Kourouthanassis, P.E., Giaglis, G.M., et Vrechopoulos, A.P. (2007). Enhancing user experience through pervasive information systems: The case of pervasive retailing. International Journal Information Management, *27(5)*, 319–335.

Kourouthanassis, P.E., Giaglis, G.M., et Karaiskos, D.C. (2008). Delineating the Degree of "Pervasiveness" in Pervasive Information Systems: An Assessment Framework and Design Implications. In Panhellenic Conference on Informatics (PCI), pp. 251–255.

Lassila, O., et Swick, R.R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation. Disponible sur http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

Lausen, H., Polleres, A., Roman, D., de Bruijn, J., et Fensel, D. (2005). Web Service Modeling Ontology (WSMO). W3C Member Submission. Disponible sur http://www.w3.org/Submission/WSMO/

Lemlouma, T. (2004). Architecture de Négociation et d'Adaptation de Services Multimédia dans des Environnements Hétérogènes. Thèse de doctorat, Institut National Polytechnique de Grenoble - INPG.

Liu, J., et Issarny, V. (2004). QoS-aware service location in mobile ad hoc networks. In IEEE International Conference on Mobile Data Management, pp. 224–235.

M'bareck, N.O.A., et Tata, S. (2008). Services Web : revue des approches de description sémantique. In Système d'Information et Intelligence Economique (SIIE)

Ma, Z., Liu, L., Yang, H., et Mylopoulos, J. (2011). Adaptive Service Composition Based on Runtime Requirements Monitoring. In IEEE International Conference on Web Services (ICWS), pp. 339–346.

Maamar, Z., Benslimane, D., et Narendra, N.C. (2006). What can context do for Web services? Communication of ACM *49(12)*, 98–103.

MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., et Metz, J. (2006). Reference Model for Service Oriented Architecture 1.0. OASIS Standard. Disponible sur http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html

Manola, F., Miller, E., et McBride, B. (2004). RDF Primer. W3C recommendation. Disponible sur http://www.w3.org/TR/rdf-primer/

Martin, D., Hayes, P., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al. (2004). OWL-S: Semantic Markup for Web Services. W3C Member Submission. Disponible sur http://www.w3.org/Submission/OWL-S/

Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Mcguinness, D.L., Sirin, E., et Srinivasan, N. (2007). Bringing Semantics to Web Services with OWL-S. Journal World Wide Web *10(3)*, pp. 243–277.

Martinetz, T.M., Berkovich, S.G., et Schulten, K.J. (1993). `Neural-gas' network for vector quantization and its application to time-series prediction. IEEE Transactions on Neural Networks, *4(4)*, pp. 558–569.

Masmoudi, N., et Conan, D. (2013). Contrats de contexte pour la gestion de contexte répartie. In Actes Des 9ème Journées Francophones : Mobilité et Ubiquité 2013 (UbiMob'13),.

Mayrhofer, R. (2004). An Architecture for Context Prediction. Thèse de doctorat, Johannes Kepler University.

McIlraith, S.A., Son, T.C., et Zeng, H. (2001). Semantic Web Services. IEEE Intelligent System, *16(2)*, pp. 46–53.

Meiners, M., Zaplata, S., et Lamersdorf, W. (2010). Structured Context Prediction: A Generic Approach. In Distributed Applications and Interoperable Systems, F. Eliassen, and R. Kapitza, eds. (Springer Berlin Heidelberg), pp. 84–97.

Minoli, D. (2008). Enterprise architecture A to Z: frameworks, business process modeling, SOA, and infrastructure technology. (Boca Raton: CRC Press), 512 p.

Mirbel, I., et Crescenzo, P. (2009). Improving Collaborations in Neuroscientist Community. In IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies (WI-IAT), pp. 567–570.

Mirbel, I., et Crescenzo, P. (2010a). Des besoins des utilisateurs à la recherche de services Web : une approche sémantique guidée par les intentions. Ingénierie des Systèmes d'Information, *15(4)*, pp. 89–112.

Mirbel, I., et Crescenzo, P. (2010b). From End-User's Requirements to Web Services Retrieval: A Semantic and Intention-Driven Approach. In Exploring Services Science, J.-H. Morin, J. Ralyté, and M. Snene, eds. (Springer Berlin Heidelberg), pp. 30–44.

Mitra, N., et Lafon, Y. (2007). SOAP Version 1.2 Part 0: Primer. W3C Recommendation. Disponible sur http://www.w3.org/TR/soap12-part0/

Ben Mokhtar, S., Preuveneers, D., Georgantas, N., Issarny, V., et Berbers, Y. (2008). EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support. Journal of System and Software, *81(5)*, pp. 785–808.

Mostefaoui, G.K., Pasquier-Rocha, J., et Brezillon, P. (2004). Context-aware computing: a guide for the pervasive computing community. In Proceedings of the IEEE/ACS International Conference on Pervasive Services (ICPS), pp. 39–48.

Musolesi, M. (2011). Socially-aware Systems. In Pervasive Adaptation : The Next Generation Pervasive Computing Research Agenda, A. Ferscha, ed. (Inst. for Pervasive Computing, Johannes Kepler Univ.), pp. 74–75.

Najar, S., Saidani, O., Kirsch-Pinheiro, M., Souveyet, C., et Nurcan, S. (2009). Semantic representation of context models : a framework for analyzing and understanding. In Proceedings of the 1st Workshop on Context, Information and Ontologies (CIAO) - 6th European Semantic Web Conference (ESWC), (New York, ACM), pp. 1-10.

Najar, S., Pinheiro, M.K., et Souveyet, C. (2011a). Bringing context to intentional services. In the Third International Conferences on Advanced Service Computing (Service Computation), Italie, pp. 118-123.

Najar, S., Pinheiro, M.K., et Souveyet, C. (2011b). Towards Semantic Modeling of intentional pervaisve System. In In 6th International Workshop on Enhanced Web Service Technologies (WEWST) - European Conference on Web Services (ECOWS), Suisse, pp. 30–34.

Najar, S., Kirsch-Pinheiro, M., et Souveyet, C. (2011c). The Influence of Context on Intentional Service. In 5th International IEEE Workshop on Requirements Engineerings for Services (REFS) – IEEE conference on Computers, Software and Applications (COMPSAC), pp. 470–475.

Najar, S., Kirsch Pinheiro, M., et Souveyet, C. (2012a). Enriched Semantic Service Description for Service Discovery : Bringing Context to Intentional Services. International Journal on Advances in Intelligent Systems *5()*, 159–174.

Najar, S., Kirsch-Pinheiro, M., Souveyet, C., et Steffenel, L.A. (2012b). Service Discovery Mechanism for an Intentional Pervasive Information System. In IEEE 19th International Conference on Web Services (ICWS), pp. 520–527.

Najar, S., Kirsch Pinheiro, M., Steffenel, L.A., et Souveyet, C. (2012c). Analyse des mécanismes de découverte de services avec prise en charge du contexte et de l'intention. In 8èmes Journées Francophones Mobilité et Ubiquité (Ubimob), (Anglet, France : Cépaduès Editions), pp. 210–221.

Najar, S., Kirsch Pinheiro, M., et Souveyet, C. (2012d). Mécanisme de prédiction dans un système d'information pervasif et intentionnel. In 8èmes Journées Francophones Mobilité et Ubiquité (Ubimob), (Anglet, France : Cépaduès Editions), pp. 146–157.

Najar, S., Pinheiro, M.K., Grand, B.L., et Souveyet, C. (2013a). Systèmes d'Information Pervasifs et Espaces de Services : Définition d'un cadre conceptuel. In 9èmes Journées Francophones Mobilité et Ubiquité (UbiMob), Nancy, sciencesconf.org :ubimob2013 :19119.

Najar, S., Pinheiro, M.K., Vanrompay, Y., Steffenel, L.A., et Souveyet, C. (2013b). Intention Prediction Mechanism In An Intentional Pervasive Information System. In Intelligent Technologies

and Techniques for Pervasive Computing, K. Kolomvatsos, C. Anagnostopoulos, and S. Hadjiefthymiades, eds. (IGI Global), pp. 251–275.

Nelles, O. (2001). Nonlinear System Identification : From Classical Approaches to Neural Networks and Fuzzy Models (Springer), 808 p.

Newcomer, E. (2002). Understanding Web Services: XML, WSDL, SOAP, and UDDI (Addison-Wesley Professional), 368 p.

Noy, N.F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R.W., et Musen, M.A. (2001). Creating Semantic Web contents with Protege-2000. Journal IEEE Intelligent Systems, *16(2)*, pp. 60–71.

Nurcan, S. (2012). Ingénierie et Architecture d'Entreprise et des Systèmes d'Information : Concepts, Fondements et Méthodes. Habilitation à Diriger la recherche, Université Panthéon-Sorbonne Paris 1.

O'Hare, G., et O'Grady, M. (2002). Addressing Mobile HCI Needs through Agents. In Human Computer Interaction with Mobile Devices, F. Paternò, ed. (Springer Berlin Heidelberg), pp. 311–314.

O'Sullivan, J., Edmond, D., Hofstede, A. ter, Properties, S., Benatallah, R.B., et Casati, F. (2002). What's in a Service? Towards Accurate Description of Non-Functional Service Properties. In Distributed Parallel Databases

OASIS (2001). UDDI White Papers. Disponible sur http://uddi.xml.org/node/33

Olsson, T., Bjurling, B., Chong, M.Y., et Ohlman, B. (2011). Goal Refinement for Automated Service Discovery. In Proceedings of the Third International Conferences on Advanced Service Computing, Rome, Italy, pp. 46–51.

OWL-S API (2012). Disponible sur http://on.cs.unibas.ch/owls-api/.version 3.1.

Paolucci, M., Kawamura, T., Payne, T.R., et Sycara, K. (2002). Semantic Matching of Web Services Capabilities. In The Semantic Web (ISWC), I. Horrocks, and J. Hendler, eds. (Springer Berlin Heidelberg), pp. 333–347.

Papazoglou, M.P. (2003). Service-oriented computing : concepts, characteristics and directions. In Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE 2003), pp. 3–12.

Papazoglou, M.P., et Georgakopoulos, D. (2003). Introduction : Service-oriented computing. Communication of the ACM *46(10)*, 24–28.

Papazoglou, M.P., et Heuvel, W.-J. (2007). Service oriented architectures : approaches, technologies and research issues. International Journal on Very Large Data Bases (VLDB), *16*(3), pp. 389–415.

Papazoglou, M.P., Traverso, P., Dustdar, S., et Leymann, F. (2008). Service-Oriented Computing : A Research Roadmap. International Journal of Cooperative Information Systems, *17(2)*

Parsia, B., et Sirin, E. (2004). Pellet: An owl dl reasoner. In Proceedings of the International Workshop on Description Logics.

Pascoe, J. (1998). Adding generic contextual capabilities to wearable computers. In Second International Symposium on Wearable Computers, Digest of Papers, pp. 92–99.

Paspallis, N., Rouvoy, R., Barone, P., Papadopoulos, G.A., Eliassen, F., et Mamelli, A. (2008). A Pluggable and Reconfigurable Architecture for a Context-Aware Enabling Middleware System. In On the Move to Meaningful Internet Systems: OTM 2008, R. Meersman, and Z. Tari, eds. (Springer Berlin Heidelberg), pp. 553–570.

Paspallis, N. (2009). Middleware-base development of context-aware applications with reusable componenets. Thèse de doctorat, University of Cyprus.

Patel-Schneider, P.F., Hayes, P., et Horrocks, I. (2004). OWL Web Ontology Language Semantics and Abstract Syntax. W3C recommendation. Disponible sur http://www.w3.org/TR/owl-semantics/

Penserini, L., Perini, A., Susi, A., et Mylopoulos, J. (2007). High variability design for software agents: Extending Tropos. ACM Transactions on Autonomous and Adaptive Systems (TAAS), *2(4)*.

Petit, M. (2010). Approche spatiale pour la caractérisation du contexte d'éxécution d'un système d'information ubiquitaire. Thèse de doctorat, Ecole nationale supérieure d'arts et métiers.

Plihon, V., Ralyte, J., Benjamen, A., Maiden, N.A.., Sutcliffe, A., Dubois, E., et Heymans, P. (1998). A reuse-Oriented Approach for the Construction of Scenario Bases Methods. In Proceedings of International Conference on Software Process, (États-Unis), pp. 1–16.

Pope, A.L. (1998). The CORBA reference guide : understanding the Common Object Request Broker Architecture (Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.), 407 p.

Prat, N. (1997). Goal Formalisation and Classification for Requirements Engineering. In Proceedings of the third International Workshop on Requirements Engineering : Foundation for Software Quality.

Preuveneers, D. (2009). Support for context-driven applications in Ambient Intelligence environments. Thèse de doctorat, Katholieke Universiteit Leuven.

Preuveneers, D., et Berbers, Y. (2010). Context-driven migration and diffusion of pervasive services on the OSGi framework. International Journal of Autonomous and Adaptive Communications Systems, *3(1)*, pp. 3–22.

Preuveneers, D., Bergh, J.V.D., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, E., Coninx, K., et Bosschere, K.D. (2004). Towards an extensible context ontology for ambient intelligence. In Proceedings of the Second European Symposium on Ambient Intelligence, (Springer-Verlag), pp. 148–159.

Preuveneers, D., Victor, K., Vanrompay, Y., Rigole, P., Kirsch-Pinheiro, M., et Berbers, Y. (2009). Context-aware adaptation in an ecology of applications. In ContextAware Mobile and Ubiquitous Computing for Enhanced Usability: Adaptive Technologies and Applications, D. Stojanovic, ed. (IGI Global)

Rabiner, L.R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. In Proceedings of the IEEE, *77(2)*, pp. 257–286.

Ramadour, P., et Fakhri, M. (2011). Modèle et langage de composition de services. In Actes Du XXIXème Congrès INFORSID, (lille), pp. 59–76.

Ranganathan, A., et Campbell, R.H. (2003). A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. In Middleware 2003, M. Endler, and D. Schmidt, eds. (Springer Berlin Heidelberg), pp. 143–161.

Redmond, F.E. (1997). Dcom: Microsoft Distributed Component Object Model with Cdrom (Foster City, CA, USA: Hungry Minds Inc), 400 p.

Reenskaug, T. (2003). The Model-View-Controller (MVC) Its Past and Present. In Java Zone, (Oslo).

Reichle, R., Wagner, M., Khan, M.U., Geihs, K., Lorenzo, J., Valla, M., Fra, C., Paspallis, N., et Papadopoulos, G.A. (2008). A Comprehensive Context Modeling Framework for Pervasive Computing Systems. In Distributed Applications and Interoperable Systems, R. Meier, and S. Terzis, eds. (Springer Berlin Heidelberg), pp. 281–295.

Reix, R. (2004). Systèmes d'information et management des organisations (Vuibert), 480 p.

Rey, G., et Coutaz, J. (2004). Le contexteur : capture et distribution dynamique d'information contextuelle. In Actes de La Première Journée Francophones : Mobilité et Ubiquité (UbiMob), (Nice, France: ACM), pp. 131–138.

Rodden, T., Chervest, K., Davies, N., et Dix, A. (1998). Exploiting Context in HCI Design for Mobile Systems. In First Workshop on Human Computer Interaction with Mobile Devices.

Rolland, C. (2007). Capturing System Intentionality with Maps. In Conceptual Modelling in Information Systems Engineering, J. Krogstie, A.L. Opdahl, and S. Brinkkemper, eds. (Berlin, Heidelberg: Springer Berlin Heidelberg), pp. 141–158.

Rolland, C., Souveyet, C., et Achour, C.B. (1998). Guiding goal modeling using scenarios. IEEE Transactions on Software Engineering, *24(12)*, pp. 1055–1071.

Rolland, C., Souveyet, C., et Kraiem, N. (2008). An Intentional view of Service Oriented Computing. Ingénierie des Systèmes d'Information, *13(1)*, pp. 107–137.

Rolland, C., Kirsch-Pinheiro, M., et Souveyet, C. (2010). An Intentional Approach to Service Engineering. IEEE Transactions on Services Computing, *3(4)*, pp. 292–305.

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web Service Modeling Ontology. Journal Applied Ontology, *1(1)*, pp. 77–106.

Römer, K., et Friedemann, M. (2010). Adaptation without anticipation. In Pervasive Adaptation : The Next Generation Pervasive Computing Research Agenda, A. Ferscha, ed. (Inst. for Pervasive Computing, Johannes Kepler Univ.), pp. 28–29.

Roshen, W. (2009). SOA-Based Enterprise Integration : A Step-by-Step Guide to Services-based Application (McGraw-Hill Osborne Media), 384 p.

Rouvoy, R., Conan, D., et Seinturier, L. (2008). Software Architecture Patterns for a Context-Processing Middleware Framework. IEEE Distributed Systems Online, *9(6)*, pp. 1–1.

Ryan, N., Pascoe, J., et Morse, D. (1997). Enhanced reality fieldwork: the context-aware archaeologist assistant. In Computer Applications & Quantitative Methods in Archaeology, Exon, ed.

Saadon, N.A., et Mohamad, R. (2011). WSMO-M: NFP-aware Web service discovery for mobile computing. In Software Engineering (MySEC), In 5th Malaysian Conference, pp. 70–75.

Salber, D., Dey, A.K., et Abowd, G.D. (1999). The Context Toolkit: Aiding the Development of Context-Enabled Applications. In the Proceedings of the SIGCHI conference on Human Factors in Computing Systems, pp. 434-441.

Santos, L.O.B. da S., Ferreira Pires, L., et Sinderen van, M.J. (2008). A Goal-Based Framework for Dynamic Service Discovery and Composition. M.J. Sinderen, ed. (Portugal: INSTICC Press), pp. 67–78.

Santos, L.O.B. da S., Guizzardi, G., Pires, L.F., et Sinderen, M. (2009). From User Goals to Service Discovery and Composition. In Proceedings of the ER 2009 Workshops (CoMoL, ETheCoM, FP-UML, MOST-ONISW, QoIS, RIGiM, SeCoGIS) on Advances in Conceptual Modeling - Challenging Perspectives, (Berlin, Heidelberg: Springer-Verlag), pp. 265–274.

Savidis, A. (2010). Adaptive automations for pervasives services. In Pervasive Adaptation: The Next Generation Pervasive Computing Research Agenda, A. Ferscha, ed. (Inst. for Pervasive Computing, Johannes Kepler Univ.), p. 11.

Schilit, B.N., et Theimer, M.M. (1994). Disseminating active map information to mobile hosts. IEEE Network : The Magazine of Global Internetworking, *8(5)*, pp. 22–32.

Schilit, B.N., Adams, N., et Want, R. (1994). Context-Aware Computing Applications. In Proceedings of the First Workshop on mobile computing systems and applications, (WMCSA), pp. 85–90.

Schmidt, A. (2010). Ubiquitous Computing: Are We There Yet? Computers, *43(2)*, pp. 95–97.

Schmidt, A., Beigl, M., et Gellersen, H.-W. (1999). There is more to context than location. Computers and Graphics, *23(6)*, pp. 893–901.

Schulthess, S. (2011). Construction of a registry for searching Web service. Mémore de Master, EFREI Engineering School Paris.

Semwebcentral (2005). wsdl2owl-s. Disponible sur http:// www.semwebcentral.org/ projects/ wsdl2owl-s/

Sigg, S. (2008). Development of a novel context prediction algorithm and analysis of context prediction schemes. Thèse de doctorat, kassel university.

Sigg, S., Haseloff, S., et David, K. (2010). An Alignment Approach for Context Prediction Tasks in UbiComp Environments. IEEE Pervasive Computing, *9(4)*, pp. 90–97.

Smith, M.K., Welty, C., et Deborah L, M. (2004). OWL Web Ontology Language Guide. W3C Recommendation. Disponible sur http://www.w3.org/TR/owl-guide/

Soylu, A., De Causmaecker, P., et Desmet, P. (2009). Context and Adaptivity in Context-Aware Pervasive Computing Environments. In Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing (UIC-ATC), pp. 94–101.

Strang, T., et Linnhoff-Popien, C. (2004). A Context Modeling Survey. In Workshop on Advanced Context Modelling, Reasoning and Management (UbiComp) - The Sixth International Conference on Ubiquitous Computing, Nottingham, England

Strang, T., Linnhoff-Popien, C., et Frank, K. (2003). CoOL: A Context Ontology Language to enable Contextual Interoperability. In Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003), LNCS 2893, Paris/France, (Springer Verlag), pp. 236–247.

Super Project (2006). Super : Semantics Utilised for Process Management within et between Entreprises. Disponible sur http://www.ontotext.com/research/super

Suraci, V., Mignanti, S., and Aiuto, A. (2007). Context-aware Semantic Service Discovery. In Mobile and Wireless Communications Summit, 16th IST, pp. 1–5.

Sycara, K., Paolucci, M., Ankolekar, A., et Srinivasan, N. (2003). Automated discovery, interaction and composition of Semantic Web services. Web Semantics : Science, Services, Agents on the World Wide Web *1(1)*, pp. 27–46.

Tamminen, S., Oulasvirta, A., Toiskallio, K., et Kankainen, A. (2004). Understanding mobile contexts. Personal and Ubiquitous Computing, *8(2)*, pp. 135–143.

Toninelli, A., Corradi, A., et Montanari, R. (2008). Semantic-based discovery to support mobile context-aware service access. Computing Communications, *31(5)*, pp. 935–949.

Truong, H., and Dustdar, S. (2009). A Survey on Context-aware Web Service Systems. Internationa Journal Web Information Systems, *5(1)*, pp. 5–31.

Uschold, M., Gruninger, M., Uschold, M., et Gruninger, M. (1996). Ontologies: Principles, methods and applications. Knowedge Engineering Review, *11(2)*, pp. 93–136.

Van Der Aalst, W.M.P., Beisiegel, M., Van Hee, K.M., Konig, D., et Stahl, C. (2007). An SOA-based architecture framework. International Journal of Business Process Integration and Management, *2(2)*, pp. 91–101.

Van Lamsweerde, A. (2000). Requirements engineering in the year 00 : a research perspective. In Proceedings of the 22nd International Conference on Software Engineering, (New York, NY, USA: ACM), pp. 5–19.

Van Lamsweerde, A. (2001). Goal-oriented requirements engineerin g: a guided tour. In Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, pp. 249–262.

Vanrompay, Y. (2011). Efficient Prediction of Future Context for Proactive Smart Systems. Thèse de Doctorat, Ka tholieke Universiteit Leuven.

Vanrompay, Y., Kirsch-Pinheiro, M., et Berbers, Y. (2011). Service Selection with Uncertain Context Information. In Service-Oriented Systems and Non-Functional Properties: Future Directions, S. Reiff-Marganiec, and M. Tilly, eds. (IGI Global), pp. 192–215.

Varshavsky, A., et Patel, S. (2009). Location in Ubiquitous Computing. In Ubiquitous Computing Fundamentals, J. Krumm, ed. pp. 285–319.

Villalonga, C., Bauer, M., Huang, V., Bernat, J., et Barnaghi, P. (2010). Modeling of sensor data and context for the Real World Internet. In 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM), pp. 1–6.

W3C (2001). W3C Semantic Web Activity. Disponible sur http://www.w3.org/2001/sw/

Wang, X.H., Zhang, D.Q., Gu, T., et Pung, H.K. (2004). Ontology based context modeling and reasoning using OWL. In Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 18–22.

WAP FORUM (2006). UAProf User Agent Profile.

Weiser, M. (1991). The Computer for the Twenty-First Century. Scientific American Ubicomp Paper after Sci Am editing, *265(3)*, pp. 94–104.

Van Welie, M., et De Ridder, G. (2001). Designing for mobile devices : A context-oriented approach. In Proceedings of the IBC Conference on Usability for Mobile Devices, pp. 9–11.

Wolisz, A. (2010). Approaches for context creation and handling. In Pervasive Adaptation : The Next Generation Pervasive Computing Research Agenda, A. Ferscha, ed. (Inst. for Pervasive Computing, Johannes Kepler Univ.), pp. 64–65.

Xiao, H., Zou, Y., Ng, J., et Nigul, L. (2010). An Approach for Context-Aware Service Discovery and Recommendation. In 2010 IEEE International Conference on Web Services (ICWS), pp. 163–170.

Xin, Y., et Hao, W. (2012). A review of Web service selection process based on Qos. In IEEE International Conference on Computer Science and Automation Engineering (CSAE), pp. 486–490.

Yang, S.J.H., et Shao, N.W.Y. (2007). Enhancing pervasive Web accessibility with rule-based adaptation strategy. Expert Systems with Applications, *32(4)*, pp. 1154–1167.

Yu, E.S.-K. (1996). Modelling strategic relationships for process reengineering. Thèse de doctorat, University of Toronto.

Yu, E.S.K., et Mylopoulos, J. (1998). Why Goal-Oriented Requirements Engineering. International Working Conference on Requirement Engineering : Foundation for Software Quality,, pp.15–22.

Yu, K., Zhang, B., Zhu, H., Cao, H., et Tian, J. (2012). Towards Personalized Context-Aware Recommendation by Mining Context Logs through Topic Models. In Advances in Knowledge Discovery and Data Mining, P.-N. Tan, S. Chawla, C.K. Ho, and J. Bailey, eds. (Springer Berlin Heidelberg), pp. 431–443.

Zaremski, A.M., et Wing, J.M. (1995). Signature matching : a tool for using software libraries. ACM Transactions on Software Engineering and Methodology (TOSEM), *4(2)*, pp. 146–170.

ANNEXES

Annexe A : Extension de OWL-S (OWL-SIC)

A.1 Extension de l'ontologie du Profil du service de OWL-S pour inclure les pointeurs vers les descriptions intentionnelles et contextuelles du service

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.1/Profile.owl">
  <!ENTITY intention "http://www.citypassenger.com/services/Intention.owl">
  <!ENTITY DEFAULT "http://www.citypassenger.com/services/ExtendedProfile.owl">
<rdf:RDF
  xmlns:rdf=       "&rdf;#"
  xmlns:rdfs=      "&rdfs;#"
  xmlns:owl=       "&owl;#"
  xmlns:xsd=       "&xsd;#"
  xmlns:profile="&profile;#"
  xmlns:intention="&intention;#"
  xml:base=        "&DEFAULT;">

  <!--  ############ ########## ############ ############ ########## -->
<owl:Ontology rdf:about="">
    <owl:imports>
                    <owl:Ontology rdf:about="&profile;" />
                    <owl:Ontology rdf:about="&intention;" />
    </owl:imports>
  </owl:Ontology>

  <!--  ############ ########## ############ ############ ########## -->

  <!-- The property context indicates the external URI to the context description of the service -->
  <owl:DatatypeProperty rdf:ID="context">
    <rdfs:domain rdf:resource="&profile;#Profile"/>
    <rdfs:range rdf:resource="&xsd;#anyURI"/>
  </owl:DatatypeProperty>

<!-- The property has_intention is a pointer to the intention that is associated with the service -->
  <owl:ObjectProperty rdf:ID="has_intention">
    <rdfs:domain rdf:resource="&profile;#Profile"/>
    <rdfs:range rdf:resource="&intention;#Intention"/>
  </owl:ObjectProperty>
  <owl:FunctionalProperty rdf:about="#has_intention"/>

</rdf:RDF>
```

A.2 L'ontologie d'intention dans OWL-SIC

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY Intention "http://www.semanticweb.org/ontologies/2011/2/Intention.owl#" >
]>

<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/2011/2/Intention.owl#"
     xml:base="http://www.semanticweb.org/ontologies/2011/2/Intention.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:Intention="http://www.semanticweb.org/ontologies/2011/2/Intention.owl#">
    <owl:Ontology rdf:about=""/>

    <!-- ///////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////->

<!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#hasParameter -->

    <owl:ObjectProperty rdf:about="#hasParameter">
        <rdfs:domain rdf:resource="#Intention"/>
        <rdfs:range rdf:resource="#Parameter"/>
    </owl:ObjectProperty>

    <!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#hasTarget -->

    <owl:ObjectProperty rdf:about="#hasTarget">
        <rdfs:domain rdf:resource="#Intention"/>
        <rdfs:range rdf:resource="#Target"/>
    </owl:ObjectProperty>

    <!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#hasVerb -->

    <owl:ObjectProperty rdf:about="#hasVerb">
        <rdfs:domain rdf:resource="#Intention"/>
        <rdfs:range rdf:resource="#Verb"/>
    </owl:ObjectProperty>

    <!-- ///////////////////////////////////////////////////////////////////////////////////
    //
```

```
// Classes
//
///////////////////////////////////////////////////////////////////////////////-->

<!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Beneficiary -->

<owl:Class rdf:about="#Beneficiary">
    <rdfs:subClassOf rdf:resource="#Parameter"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Destination -->

<owl:Class rdf:about="#Destination">
    <rdfs:subClassOf rdf:resource="#Direction"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Direction -->

<owl:Class rdf:about="#Direction">
    <rdfs:subClassOf rdf:resource="#Parameter"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Intention -->

<owl:Class rdf:about="#Intention">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Manner -->

<owl:Class rdf:about="#Manner">
    <rdfs:subClassOf rdf:resource="#ways"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Means -->

<owl:Class rdf:about="#Means">
    <rdfs:subClassOf rdf:resource="#ways"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Object -->

<owl:Class rdf:about="#Object">
    <rdfs:subClassOf rdf:resource="#Target"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Parameter -->

<owl:Class rdf:about="#Parameter">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>

<!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Quality -->
```

```
    <owl:Class rdf:about="#Quality">
        <rdfs:subClassOf rdf:resource="#Parameter"/>
    </owl:Class>

    <!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Quantity -->

    <owl:Class rdf:about="#Quantity">
        <rdfs:subClassOf rdf:resource="#Parameter"/>
    </owl:Class>

    <!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Result -->

    <owl:Class rdf:about="#Result">
        <rdfs:subClassOf rdf:resource="#Target"/>
    </owl:Class>

    <!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Source -->

    <owl:Class rdf:about="#Source">
        <rdfs:subClassOf rdf:resource="#Direction"/>
    </owl:Class>

    <!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Target -->

    <owl:Class rdf:about="#Target">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#Verb -->

    <owl:Class rdf:about="#Verb">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>

    <!-- http://www.semanticweb.org/ontologies/2011/2/Intention.owl#ways -->

    <owl:Class rdf:about="#ways">
        <rdfs:subClassOf rdf:resource="#Parameter"/>
    </owl:Class>


    <!-- http://www.w3.org/2002/07/owl#Thing -->

    <owl:Class rdf:about="&owl;Thing"/>
</rdf:RDF>
```

A.3 Extension du Modèles de Processus du service de OWL-S pour inclure la variabilité et la composition intentionnelle et contextuelle du service

```
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY shadow-rdf "http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl">
  <!ENTITY expr "http://www.daml.org/services/owl-s/1.1/generic/Expression.owl">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY time "http://www.isi.edu/~pan/damltime/time-entry.owl">
  <!ENTITY swrl "http://www.w3.org/2003/11/swrl">
  <!ENTITY service "http://127.0.0.1/ontology/Service.owl">
  <!ENTITY grounding "http://127.0.0.1/ontology/Grounding.owl">
  <!ENTITY process "http://127.0.0.1/ontology/Process.owl">
  <!ENTITY DEFAULT "http://127.0.0.1/ontology/ExtendedProcess.owl">
]>

<rdf:RDF
  xmlns:rdf=        "&rdf;#"
  xmlns:shadow-rdf= "&shadow-rdf;#"
  xmlns:expr= "&expr;#"
  xmlns:rdfs=       "&rdfs;#"
  xmlns:owl= "&owl;#"
  xmlns:swrl= "&swrl;#"
  xmlns:xsd= "&xsd;#"
  xmlns:service= "&service;#"
  xmlns:process= "&process;#"
  xmlns:grounding= "&grounding;#"
  xmlns=              "&DEFAULT;#"
  xml:base="&DEFAULT;;">

  <!--  ############ ########## ############ ############ ########## -->

  <owl:Ontology rdf:about="">
    <owl:imports>
          <owl:Ontology rdf:about="&process;" />
    </owl:imports>
  </owl:Ontology>

  <!--  ############ ########## ############ ############ ########## -->

<!--#########################################################
              Context Condition
   ##########################################################-->

<owl:Class rdf:ID="ContextCondition">
  <rdfs:subClassOf rdf:resource="#Parameter"/>
</owl:Class>

<!--#########################################################
    Intentional Atomic, Simple and aggregate Processes
```

```
   ###########################################################-->

<owl:Class rdf:ID="IntentionalAtomicProcess">
  <rdfs:subClassOf rdf:resource="#Process"/>
</owl:Class>

<owl:Class rdf:ID="IntentionalSimpleProcess">
  <rdfs:subClassOf rdf:resource="#Process"/>
  <owl:disjointWith rdf:resource="#IntentionalAtomicProcess"/>
</owl:Class>

<owl:Class rdf:ID="IntentionalAggregateProcess">
  <rdfs:subClassOf rdf:resource="#Process"/>
  <owl:disjointWith rdf:resource="#IntentionalAtomicProcess"/>
  <owl:disjointWith rdf:resource="#IntentionalSimpleProcess"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="realizedBy">
  <rdfs:domain rdf:resource="#IntentionalSimpleProcess"/>
  <rdfs:range rdf:resource="#IntentionalAtomicProcess"/>
  <owl:inverseOf rdf:resource="#realizes"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="realizes">
  <rdfs:domain rdf:resource="#IntentionalAtomicProcess"/>
  <rdfs:range rdf:resource="#IntentionalSimpleProcess"/>
  <owl:inverseOf rdf:resource="#realizedBy"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="expandsTo">
  <rdfs:domain rdf:resource="#IntentionalSimpleProcess"/>
  <rdfs:range rdf:resource="#IntentionalAggregateProcess"/>
  <owl:inverseOf rdf:resource="#collapsesTo"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="collapsesTo">
  <rdfs:domain rdf:resource="#IntentionalAggregateProcess"/>
  <rdfs:range rdf:resource="#IntentionalSimpleProcess"/>
  <owl:inverseOf rdf:resource="#expandsTo"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="expand">
  <rdfs:comment>This is a deprecated usage; expandsTo is preferred.</rdfs:comment>
  <owl:equivalentProperty rdf:resource="#expandsTo"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="collapse">
  <rdfs:comment>This is a deprecated usage; collapsesTo is preferred.</rdfs:comment>
  <owl:equivalentProperty rdf:resource="#collapsesTo"/>
</owl:ObjectProperty>

<!--#########################################################################
   Intentional Composite Processes and Intentional Composition Constructs
```

```
   ##########################################################################-->

<!-- ..... Intentional Composite Processes ..... -->

<owl:Class rdf:ID="IntentionalCompositeProcess">
  <rdfs:subClassOf rdf:resource="#IntentionalAggregateProcess"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="composedOf">
  <rdfs:domain rdf:resource="#IntentionalCompositeProcess"/>
  <rdfs:range rdf:resource="#IntentionalCompositionConstruct"/>
</owl:ObjectProperty>

<!--  IntentionalCompositionConstruct Class-->

<owl:Class rdf:ID="IntentionalCompositionConstruct">
</owl:Class>

<owl:ObjectProperty rdf:ID="components">

  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Sequence"/>
        <owl:Class rdf:about="#Parallel"/>
        <owl:Class rdf:about="#Iterative"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>


<!-- Collections of Intentional Composition Constructs -->

<owl:Class rdf:ID="IntentionalCompositionConstructBag">
  <rdfs:comment> A multiset of intentional composition constructs </rdfs:comment>
  <rdfs:subClassOf rdf:resource="&shadow-rdf;#List"/>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&shadow-rdf;#first"/>
    <owl:allValuesFrom rdf:resource="#IntentionalCompositionConstruct"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&shadow-rdf;#rest"/>
    <owl:allValuesFrom rdf:resource="#IntentionalCompositionConstructBag"/>
   </owl:Restriction>
  </rdfs:subClassOf>

</owl:Class>

<!-- IntentionalCompositionConstructList.-->
```

```
<owl:Class rdf:ID="IntentionalCompositionConstructList">
<rdfs:subClassOf rdf:resource="&shadow-rdf;#List"/>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&shadow-rdf;#first"/>
    <owl:allValuesFrom rdf:resource="#IntentionalCompositionConstruct"/>
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<!--  Sequence.-->

<owl:Class rdf:ID="Sequence">
  <rdfs:subClassOf rdf:resource="#IntentionalCompositionConstruct"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#components"/>
      <owl:allValuesFrom rdf:resource="#IntentionalCompositionConstructList"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<!-- Parallel.  -->

<owl:Class rdf:ID="Prallel">

  <rdfs:subClassOf rdf:resource="#IntentionalCompositionConstruct"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#components"/>
      <owl:allValuesFrom rdf:resource="#IntentionalCompositionConstructBag"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<!-- Iteratif.-->

<owl:Class rdf:ID="Iteratif">
<rdfs:subClassOf rdf:resource="#IntentionalCompositionConstruct"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#components"/>
      <owl:allValuesFrom rdf:resource="#IntentionalCompositionConstructBag"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<!--#############################################################################
    Intentional Variable Processes and Intentional Variable Constructs
    #############################################################################-->

<!-- ..... Intentional Variable Processes ..... -->
```

```
<owl:Class rdf:ID="IntentionalVariableProcess">
  <rdfs:subClassOf rdf:resource="#IntentionalVariableProcess"/>
</owl:Class>

<owl:ObjectProperty rdf:ID="variantOf">
  <rdfs:domain rdf:resource="#IntentionalVariableProcess"/>
  <rdfs:range rdf:resource="#IntentionalVariableConstruct"/>
</owl:ObjectProperty>

<!--  IntentionalVariableConstruct Class-->

<owl:Class rdf:ID="IntentionalVariableConstruct">
</owl:Class>

<owl:ObjectProperty rdf:ID="components">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Alternative"/>
        <owl:Class rdf:about="#Multiple"/>
        <owl:Class rdf:about="#Choice"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:ObjectProperty>

<!-- Collections of Intentional Variable Constructs -->

<owl:Class rdf:ID="IntentionalVariableConstructBag">
  <rdfs:comment> A multiset of intentional Variable constructs </rdfs:comment>
  <rdfs:subClassOf rdf:resource="&shadow-rdf;#List"/>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&shadow-rdf;#first"/>
    <owl:allValuesFrom rdf:resource="#IntentionalVariableConstruct"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&shadow-rdf;#rest"/>
    <owl:allValuesFrom rdf:resource="#IntentionalVariableConstructBag"/>
   </owl:Restriction>
  </rdfs:subClassOf>

</owl:Class>

<!-- IntentionalVariableConstructList.-->

<owl:Class rdf:ID="IntentionalVariableConstructList">

  <rdfs:subClassOf rdf:resource="&shadow-rdf;#List"/>
  <rdfs:subClassOf>
```

```
   <owl:Restriction>
    <owl:onProperty rdf:resource="&shadow-rdf;#first"/>
    <owl:allValuesFrom rdf:resource="#IntentionalVariableConstruct"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&shadow-rdf;#rest"/>
    <owl:allValuesFrom rdf:resource="#IntentionalVariableConstructList"/>
   </owl:Restriction>
  </rdfs:subClassOf>

</owl:Class>

<!-- Alternative.-->

<owl:Class rdf:ID="Alternative">
  <rdfs:subClassOf rdf:resource="#IntentionalVariableConstruct"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#components"/>
      <owl:allValuesFrom rdf:resource="#IntentionalVariableConstructBag"/>
    </owl:Restriction>
  </rdfs:subClassOf>

</owl:Class>

<!-- Multiple.  -->

<owl:Class rdf:ID="Multiple">
  <rdfs:subClassOf rdf:resource="#IntentionalVariableConstruct"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#components"/>
      <owl:allValuesFrom rdf:resource="#IntentionalVariableConstructBag"/>
    </owl:Restriction>
  </rdfs:subClassOf>

</owl:Class>

<!--  Choice.  -->

<owl:Class rdf:ID="Choice">
  <rdfs:subClassOf rdf:resource="#IntentionalVariableConstruct"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#components"/>
      <owl:allValuesFrom rdf:resource="#IntentionalVariableConstructBag"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
</rdf:RDF>
```

Annexe B : Détails du cas d'étude

B.1 Description intentionnelle et contextuelle des services

Service	*Intention*	*Contexte Requis*	*Réalisation du service*
Service 1	I1 Consulter fiche client	- Dispositif.Réseau.Type≠ Ethernet - Utilisateur.Localisation.Nom ≠ Entreprise - Utilisateur.Rôle = Commercial - Utilisateur.Profil.Expertise = faible	- ***AccessClientViewVPN*** (TunnelVPN, SSLAuthentification, DataEncryption, ClientListPage, ViewClientWS)
Service 2	I1 Consulter fiche client	- Dispositif.Réseau.Type≠ Ethernet - Utilisateur.Profil.Expertise = élevé - Utilisateur.Rôle = Commercial	- ***AccessClientView*** (SSLAuthentificationWS, FindClientByAddressWS, DetailedClientView)
Service 3	I1 Consulter fiche client	- Utilisateur.Rôle = Commercial - Utilisateur.Profil.Expertise = Elevé - Dispositif.Réseau.Type≠ Ethernet	- ***AccessClientViewEncryption*** (SSLAuthentificationWS, DataEncryption, FindClientByAddressWS, DetailedClientView)
Service 4	I2 Etablir proposition	- Dispositif.Réseau.Type= Ethernet - Utilisateur.Localisation.Nom = Entreprise - Dispositif.mémoire > 512	- ***ProposalEditionFax*** (ProposalEditionWS, FaxService)
Service 5	I2 Etablir proposition	- Dispositif.Réseau.Type≠ Ethernet - Utilisateur.Localisation.Nom = Extérieur - Utilisateur.Profil.Expertise = Elevé	- ***ProposalEditionEncryptionEmail*** (AuthentificationWS, ProposalEditionWS, DataEncryption, EmailClient)

Service 6	I2 Etablir proposition	- Dispositif.Réseau.Type≠ Ethernet - Dispositif.Type ≠ Iphone - Dispositif.Mémoire >= 1024 - Utilisateur. Profil.Expertise = Faible - Utilisateur.Localisation.Nom ≠ Entreprise	- ***ProposalEditionEmail*** (TunnelVPN, AuthentificationWS, ProposalEditionWS, EmailClient)
Service 7	I3 Consulter commande	- Utilisateur.Rôle = Client - Dispositif.Réseau.Type = Ethernet - Utilisateur.Localisation.Nom = Entreprise	- ***ViewCommand*** (TraceClientCommand)
Service 8	I3 Consulter commande	- Utilisateur.Rôle = Commercial - Dispositif.Réseau.Type ≠ Ethernet - Utilisateur.Localisation.Nom ≠ Entreprise - Utilisateur.Profil.Expertise = faible	- ***ViewCommandVPN*** (TunnelVPN, AuthentificationWS, FindClientByAddressWS, TraceClientCommand)
Service 9	I4 Réaliser réunion	- Utilisateur.Rôle = Directeur - Utilisateur.Localisation.Nom ≠ Entreprise - Dispositif.Type = Intel core 2 Duo	- ***RequestHighQualityVideoConference*** (JoinVirtualRoom, StartHighQualityVideoConf)
Service 10	I4 Réaliser réunion	- Utilisateur.Rôle = Directeur - Utilisateur.Localisation.Nom = Entreprise - Dispositif.Réseau.Type= Ethernet - Dispositif.Type = Intel core 2 Duo	- ***RequestVirtualVideoConference*** (JoinVirtualRoom, StartVideoConf)
Service 11	I4 Réaliser réunion	- Utilisateur.Rôle = Technicien - Utilisateur.Localisation.Nom = Entreprise - Dispositif.Réseau.Type= Ethernet - Dispositif.Type = Intel Core 2 Duo	- ***RequestLowQualityAudioConference*** (JoinVirtualRoom, StartLowQualityAudioConf)
Service 12	I4 Réaliser réunion	- Utilisateur.Rôle = Commercial - Utilisateur.Localisation.Nom ≠ Entreprise - Dispositif.Mémoire > 512	- ***RequestVideoConference*** (RequestVirtualRoom, RequestBordWidth, StartVideoConfWS)

		- Dispositif.Réseau.Type ≠ Ethernet	
Service 13	I4 Réaliser réunion	- Utilisateur.Rôle = commercial - Utilisateur.Localisation.Nom = Entreprise - Dispositif.Réseau.Type= Ethernet - Dispositif.Type = Intel core 2 Duo	- ***RequestHighQualityVideoConference*** (RequestVirtualRoom, RequestBordWidth, StartHighQualityVideoConf)
Service 14	I4 Réaliser réunion	- Utilisateur.Rôle = Commercial - Utilisateur.Localisation.Nom ≠ Entreprise - Dispositif.Mémoire > 512 - Dispositif.Réseau.Type ≠ Ethernet - Dispositif.Type = Android	- ***RequestEncryptedLowQualityConference*** (RequestVirtualRoom, DataEncryption, StartLowQualityVideoConfWS)
Service 15	I4 Réaliser réunion	- Utilisateur.Localisation.Nom = Entreprise - Dispositif.Réseau.Type = Ethernet - Dispositif.Type = Intel Core 2 Duo - Utilisateur.Profil.Expertise = élevé	- ***RequestLowAudioConference*** (JoinVirtualRoom, StartLowQualityAudioConf)
Service 16	I4 Réaliser réunion	- Utilisateur.Localisation.Nom ≠ Entreprise - Dispositif.Réseau.Type = 3G - Dispositif.Type = Iphone - Utilisateur.Profil.Expertise = faible	- ***RequestLowVideoConferenceVPN*** (TunnelVPN, JoinVirtualRoom, StartLowQualityVideoConf)
Service 17	I5 Chercher Restaurant	- Dispositif.Réseau.Type ≠ Ethernet - Utilisateur.Rôle = Commercial - Utilisateur.Localisation.Nom ≠ Entreprise	***-FindRestaurantVPN*** (TunnelVPN, SSLAuthentification, FindNearestRestaurant, RestaurantListPage, ViewRestaurantWS)

B.2 Description des situations de l'utilisateur en termes de <Intention, Contexte, Service>

Situation	Sujet de Contexte	Espace de Services	Utilisateur		Service Sélectionné
			Intention	***Contexte***	
Situation 1	Jean	Domicile	I2 Etablir proposition	- Utilisateur.Rôle = Commercial - Utilisateur.Profil.Age = 42 - Utilisateur.Profil.Expertise = Faible - Utilisateur.Localisation.Nom = Domicile - Utilisateur.Temps = Matin - Dispositif.Type = Android 4.1 - Dispositif.Mémoire = 2048 - Dispositif.Réseau.Type= WiFi - Dispositif.Réseau.Nom = Jean-WiFi	**Service 6**
Situation 2	Jean	Entreprise	I4 Réaliser réunion	- Utilisateur.Rôle = Commercial - Utilisateur.Profil.Age = 42 - Utilisateur.Profil.Expertise = Faible - Utilisateur.Localisation.Nom = Entreprise - Utilisateur.Temps = Après-midi - Dispositif.Type = Intel Core 2 Duo - Dispositif.Mémoire = 1024 - Dispositif.Réseau.Type = Ethernet - Dispositif.Réseau.Nom = 192.168.13.141	**Service 13**
Situation 2 bis	Directeur	Entreprise	I4 Réaliser réunion	- Utilisateur.Rôle = Directeur - Utilisateur.Profil.Age = 54 - Utilisateur.Profil.Expertise = Elevé	**Service 10**

				- Utilisateur.Localisation.Nom = Entreprise - Utilisateur.Time = After-noon - Dispositif.Type = Intel Core 2 Duo - Dispositif.Mémoire = 2 Go - Dispositif.Réseau.Type = Ethernet - Dispositif.Réseau.Nom = 192.168.13.139	
Situation 3	Pierre	Endroit Public	I2 Etablir proposition	- Utilisateur.Rôle = Commercial - Utilisateur.Profil.Age = 30 - Utilisateur.Profil.Expertise = Elevé - Utilisateur.Localisation.Nom = Extérieur - Utilisateur.Temps = Après-midi - Dispositif.Type = IPad 2 - Dispositif.Réseau.Type = 3G	**Service 5**
Situation 4	Jean	Chez Client	I4 Réaliser réunion	- Utilisateur.Rôle = Commercial - Utilisateur.Profil.Age = 42 - Utilisateur.Profil.Expertise = Faible - Utilisateur.Localisation.Nom = Extérieur - Utilisateur.Temps = Après-midi - Dispositif.Type = Android 4.1 - Dispositif.Mémoire = 768 - Dispositif.Réseau.Type = WIFI - Dispositif.Réseau.Nom = FreeWifi	**Service 14**
Situation 4 Bis	Louis	Entreprise	I4 Réaliser réunion	- Utilisateur.Rôle = Technicien - Utilisateur.Profil.Age = 28 - Utilisateur.Profil.Expertise = Elevé - Utilisateur.Localisation.Nom = Entreprise - Dispositif.Type = Intel Core 2 Duo - Dispositif.Réseau.Type = Ethernet - Dispositif.Réseau.Nom = 192.168.13.145	**Service 11**

Situation 5	Jean	Endroit Public	I1 Consulter fiche client	- Utilisateur.Rôle = Commercial - Utilisateur.Profil.Age = 42 - Utilisateur.Profil.Expertise = Faible - Utilisateur.Localisation.Nom = Extérieur - Utilisateur.Temps = Après-midi - Dispositif.Type = Android 4.1 - Dispositif.Mémoire = 786 - Dispositif.Réseau.Type = WIFI - Dispositif.Réseau.Nom = FreeWifi	**Service 1**
Situation 6	Louis	Entreprise	I1 Consulter fiche client	- Utilisateur.Rôle = Technicien - Utilisateur.Profil.Age = 28 - Utilisateur.Profil.Expertise = Elevé - Utilisateur.Localisation.Nom = Entreprise - Utilisateur.Temps = Après-midi - Dispositif.Type = Intel Core 2 Duo - Dispositif.Réseau.Type = Ethernet - Dispositif.Réseau.Nom = 192.168.13.145	**Service 2**
Situation 7	Jean	Extérieur	I5 Rechercher restaurant	- Utilisateur.Rôle = Commercial - Utilisateur.Profil.Age = 42 - Utilisateur.Profil.Expertise = Faible - Utilisateur.Localisation.Nom = Extérieur - Utilisateur.Temps = Après-midi - Dispositif.Type = Android 4.1 - Dispositif.Mémoire = 786 - Dispositif.Réseau.Type = WIFI - Dispositif.Réseau.Nom = FreeWifi	**Service 17**

Annexe C : Liste de publications en relation avec ce travail de thèse

C.1 Chapitres

[1] Najar, S., Pinheiro, M.K., Vanrompay, Y., Steffenel, L.A., and Souveyet, C. (2013). Intention Prediction Mechanism In An Intentional Pervasive Information System. In Intelligent Technologies and Techniques for Pervasive Computing, K. Kolomvatsos, C. Anagnostopoulos, and S. Hadjiefthymiades, eds. (IGI Global), pp. 251–275.

C.2 Articles dans des revues à comité de lecture

[2] Najar, S., Kirsch Pinheiro, M., and Souveyet, C. (2012). Enriched Semantic Service Description for Service Discovery : Bringing Context to Intentional Services. International Journal on Advances in Intelligent Systems, *5(1-2)*, 159–174.

C.3 Communication avec actes

[3] Najar, S (2009). Adaptation dynamique des services en fonction d'un contexte utilisateur en adoptant une approche intentionnelle. In XXVIIème Congrès INFORSID : Informatique des Organisations et Systèmes d'Information et de Décision, TOULOUSE, pp. 215–223., pp. 461-462

[4] Najar, S., Saidani, O., Kirsch-Pinheiro, M., Souveyet, C., and Nurcan, S. (2009). Semantic representation of context models : a framework for analyzing and understanding. In Proceedings of the 1st Workshop on Context, Information and Ontologies (CIAO)- 6th European Semantic Web Conference (ESWC), (New York, ACM), pp. 1-10.

[5] Najar, S (2010). Context-Aware Intentional Service Framework for service adaptation. In 4th IEEE International Conference on Research Challenges in Information Science (RCIS), Nice, France, pp. 663-672

[6] Najar, S., Pinheiro, M.K., and Souveyet, C. (2011). Bringing context to intentional services. In the Third International Conferences on Advanced Service Computing (Service Computation), Italie, pp. 118-123.

[7] Najar, S., Pinheiro, M.K., and Souveyet, C. (2011). Towards Semantic Modeling of intentional pervaisve System. In 6th International Workshop on Enhanced Web Service Technologies (WEWST) - European Conference on Web Services (ECOWS), Suisse, pp. 30–34.

[8] Najar, S., Kirsch-Pinheiro, M., and Souveyet, C. (2011). The Influence of Context on Intentional Service. In 5th International IEEE Workshop on Requirements Engineerings for Services (REFS) – IEEE conference on Computers, Software and Applications (COMPSAC), pp. 470–475.

[9] Najar, S., Kirsch-Pinheiro, M., Souveyet, C., and Steffenel, L.A. (2012). Service Discovery Mechanism for an Intentional Pervasive Information System. In IEEE 19th International Conference on Web Services (ICWS), honolulu, hawaii, pp. 520–527.

[10] Najar, S., Kirsch Pinheiro, M., Steffenel, L.A., and Souveyet, C. (2012). Analyse des mécanismes de découverte de services avec prise en charge du contexte et de l'intention. In 8èmes Journées Francophones Mobilité et Ubiquité (Ubimob), (Anglet, France : Cépaduès Editions), pp. 210–221.

[11] Najar, S., Kirsch Pinheiro, M., and Souveyet, C. (2012). Mécanisme de prédiction dans un système d'information pervasif et intentionnel. In 8èmes Journées Francophones Mobilité et Ubiquité (Ubimob), (Anglet, France : Cépaduès Editions), pp. 146–157.

[12] Najar, S., Pinheiro, M.K., Grand, B.L., and Souveyet, C. (2013). Systèmes d'Information Pervasifs et Espaces de Services : Définition d'un cadre conceptuel. In 9èmes Journées Francophones Mobilité et Ubiquité (UbiMob), Nancy, sciencesconf.org : ubimob2013:19119

[13] Kirsch-Pinheiro, M., Le Grand, B., Souveyet, C., and Najar, S. (2013). Espace de Services : Vers une formalisation des Systèmes d'Information Pervasifs. In XXXIème Congrès INFORSID : Informatique des Organisations et Systèmes d'Information et de Décision, Paris, pp. 215–223.